パソコンプログラマのためのクリエイティブマガジン

平成3年4月2日第三種郵便物認可　1991年12月1日発行（毎月1回18日）第1巻第12号通巻12号

月刊プログラマーズ・ページ

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

1991 12 DECEMBER

定価980円

特集

やっぱり僕らはBASIC!

用途を選ばぬそのプログラミングパワーを探る

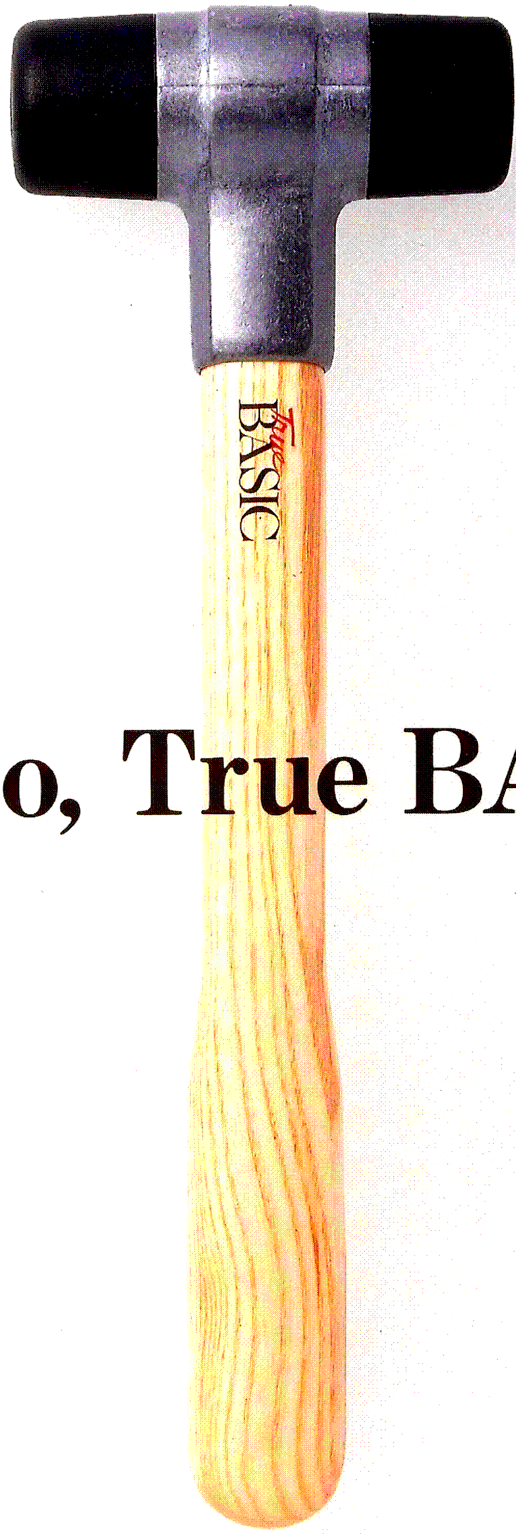
付録:5"2HDフロッピーディスク

本誌掲載プログラム　書き込み可能テキストファイル　QuickBASIC Ver.4.5ステートメント・関数一覧　クイズ・ランダムスクリーン　編集ツール　アーカイブユーティリティ「LHA」

SE
Shoeisha Co., Ltd.

創造をクレオらしさと呼びたい

CREO



Hello, True BASIC.

現代の正統派言語　選ぶなら世界標準

1964年、J.G.ケメニーとT.E.カーツの開発した教育用会話型言語「BASIC」は、コンピュータを大変身近なものにしました。すっきりした記述で初めての人でもわかりやすく、慣れればさらに高度に使いこなせる。当時としては、画期的な言語の登場です。こうして生まれ、多くの支持を得たBASICは、ケメニー、カーツ両氏の手によりさらなる進化を遂げました。それが、True BASICです。

①すっきりとわかりやすい、構造化プログラミング

True BASICは、コンピュータが初めての人にもわかりやすい構造化プログラミング。行番号はいっさい使わず、あなたの頭の中のアルゴリズムどおりに、プログラムは組まれます。基礎的な設計力が自然と身につくプログラミング法です。

②初めてでもとまどわない、一貫した操作性

例えば、フルスクリーンエディタとコンパイラとの完全統合。ソースコードの書きこみからデバッグ、コンパイルまで、作業は中断なく滑らかに進行。しかも、作成されたプログラムは直接実行ファイルへ変換でき、手間がかかりません。そのうえ、作業の見とおしがよいマルチウィンドウをサポート。テキストやグラフィックスの作成がスムーズに行なえます。

③シンプルで使いやすい、モジュール(機能)群

機能の高さだけでなく、操作性を重視したモジュール群。作業の流れを中断しない、系統立てた構成からなります。複雑な演算に強い、行列関数など豊富な関数。高速処理に威力を発揮する、数値演算コプロセッサ。よく使う実戦的機能の他にも、仮引数、再帰呼出処理といった高度な機能を満載しました。さらにグラフィックス機能が充実。各種ツールキットや組み込み関数を用いて、簡単作成できます。

④機種もOSも選ばない、優れた互換性

True BASICは、ANSI規格のFULL BASICに準拠。いわば世界標準のBASICです。そのうえ、ハードウェアに依存しない言語構造ですから、異なる機種間、MS-DOSからUNIXまでOS間の互換性があります。つまり、True BASICで構築されたプログラムは、特に変更せずに様々なマシン、OS上で活用できるということ。作成したプログラム資産を無駄にしません。

⑤True BASICは、すでに世界で活用中

対応機種	パソコン版	IBM	PC、PS/2
		Apple	Macintosh 512K、Plus SE、Macintosh II
		Atari	ST 520、1040、MegaST
		Commodore	Amiga 500、1000、2000
	UNIX版	Silicon Graphics Sun Microsystems, Inc. Data General Corp.	
採用している大学	全米で1600校以上		

多彩な機能がいろいろ選べる、ツールキット発売中!

●ビジネス用グラフィックスツールキット
価格：18,000円

棒グラフ・線グラフなどの単一グラフから、多種類のグラフを1枚の複合グラフとして表示するなど、17種類のビジネスチャートを多彩な表現で高速描画できます。

●数学者用ツールキット
価格：25,000円

10万桁の高精度な演算や複素行列の演算から、入力された関数式の評価まで、通常では大変複雑で高度な処理を強力にサポートします。

●3次元グラフィックスツールキット
価格：18,000円

3次元のウィンドウを設定し視点を定めるだけで、簡単に3次元グラフィックスが描画できます。キャビネ投影やカバリエ投影などの指定もでき、本格的な立体図形の作成も可能です。

●科学技術用グラフィックスツールキット
価格：25,000円

対数グラフや極座標グラフ、さらにベクトルグラフなど、科学技術分野では欠かせない複雑なグラフが、サブルーチンを呼ぶだけで描画できる優れたツールキットです。

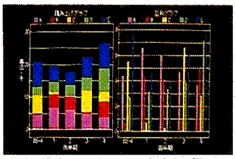
発売予定のツールキット

●文字列操作ツールキット　近日発売

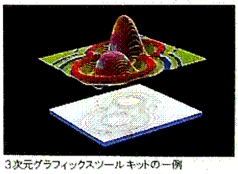
●並べ替え・検索用ツールキット　近日発売

●統計用グラフィックスツールキット　近日発売

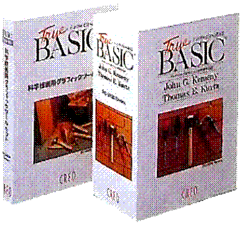
True BASIC Tool Kit



ビジネス用グラフィックスツールキットの一例



3次元グラフィックスツールキットの一例



好評発売中

True BASIC™ シンプルだから使える

PC-9800 Series

標準価格 18,000円(税別)

株式会社クレオ

〒108 東京都港区芝浦4-9-25 芝浦スクエアビル TEL03-3456-4650 FAX03-3456-4630

〈資料請求番号P02〉

Brief

Full Screen Editor 日本語版

全米のプログラマが認めた実力！
アメリカでは「ブリーフ」が
プログラミング用エディターとして
最も多く利用されています。

C言語で組めるマクロ！

v.3.0 いよいよ登場。

Version UP!

プログラミングエディター新世紀。

よりパワフルで、よりエレガントな プログラミングのために……。

●C言語で組めるマクロ 新機能

使い慣れているC言語でマクロが組めますから、習得が簡単でカスタマイズも思いのままです。

●マクロのスクリーン・デバッガ 新機能

ソースコードでデバックできます。もちろんブレークポイントやウォッチ・ドッグもセットできます。

●テンプレート機能〈構文の自動生成〉 新機能

C、Ada、Basic、COBOL、FORTRAN、MODULA2、Pascalで構文を自動生成します。

●300回ものアンドゥ

各ファイルごとに300までの操作を元にもどすことができますので、やり直しの面倒を気にせず、安心して編集できます。

●安全編集

Briefは、一定時間キーを操作しないこと最新の編集内容をファイルに自動保存します。だから編集中に電源が切れても安心です。

●その他の主な新機能

キー操作のファイルへのセーブとリプレイ、処理可能行数を拡張、リジューム機能、Ctrlキー・アサイン(オプション)、メニュー(オプション)etc。

アメリカでも日本でも高い評価

- ●プログラミング向きエディターを探している人は、もう探す必要はない。(Jerry Pournelle/BYTE)
- ●ソフト開発には強力なツール、300回のアンドゥは大きな特徴。(日経バイト)
- ●既存のエディターに不満を覚えたら、これしかない。(Oh! FM)
- ●まるで夢のようなアンドゥだ――おもわず間違うのを期待してしまいそうだ。(The C Users Journal)

■主な機能

◆55ウィンドウ◆無限のファイル◆自動コンパイル◆自動エラー・ジャンプ◆自動インデント◆自動テンプレート◆バッファ書出/切換/削除/リスト表示◆ファイル名の指定/表示/変更◆バックアップ保存◆特殊文字入力◆2行を1行に連結◆ファイル単位の挿入◆増分検索◆各種マーク/始・終点の入換◆ブロックのカット/コピー/ペースト/印刷/保存/左右へのシフト◆対応する括弧へ移動◆C言語のコメント化/非コメント化◆DOSコマンドの実行◆キー操作記憶/実行◆入力ヒストリー◆英文テキスト整形◆タブのスペース挿入◆拡張子毎に異なった設定ができるタブ◆検索置換パターンの画面からの入力◆英字の大小文字変換/その他

定価 48,000円
(消費税は含まず)

■5機種対応パッケージ

(東芝、NEC、AX協議会、IBM、富士通)

●5"2DD、3.5"2DDデュアルメディア●MS-DOS 3.1以降に対応●英語DOSでは使用できませんが、英語版BRIEFもセット販売いたしております。●お急ぎの方、上記以外のメディア希望の方、英語BRIEFのセット販売希望の方はユーザー・サポート係までご連絡下さい。

日本システム株式会社

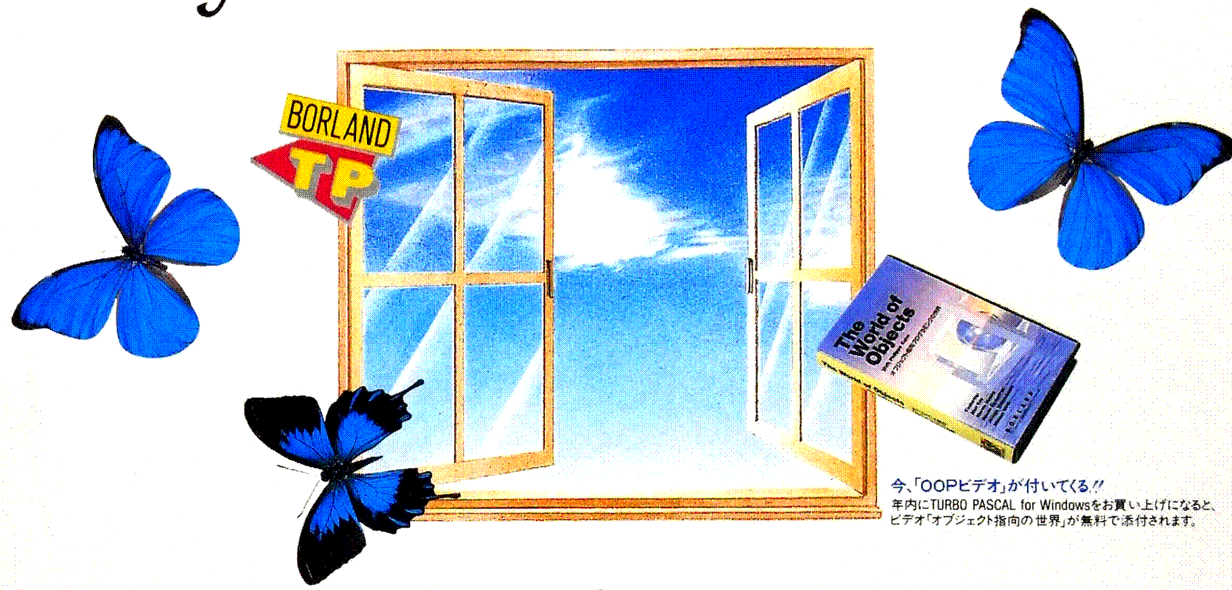
●お問合せは　東京都立川市曙町1-18-2一清ビル立川事業所Briefユーザー・サポート係
TEL0425-23-4469　FAX0425-27-6127

〈資料請求番号001〉

BORLAND

The Leader in Object-Oriented Programming

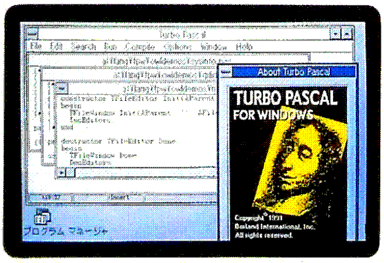
"Easy & True" for Windows.



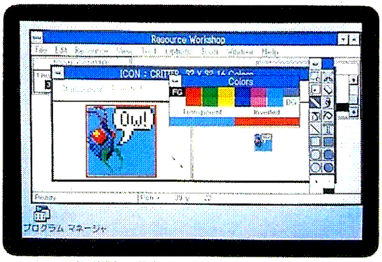
今、「OOPビデオ」が付いてくる!!
年内にTURBO PASCAL for Windowsをお買い上げになると、ビデオ「オブジェクト指向の世界」が無料で添付されます。

基本は、ターボパスカル——。
完ぺきなWindowsアプリケーション開発環境を提供。

その美しい言語仕様から"コンパイラの貴婦人"と呼ばれ、多くのプログラマーを魅了してきたパスカルが今、「Turbo Pascal for Windows」として生まれ変わりました。デビュー当時、その驚異的な低価格とコンパイルスピードの速さ、デバッガやエディタを備えた世界初の統合環境で人々を圧倒し、瞬く間に標準の地位を確立して以来、ターボパスカルは常に先進の機能を搭載し、言語の世界をリードし続けています。アメリカでは、大学などの教育機関や研究施設でターボパスカルを使用することが常識ですし、ヨーロッパや日本でもビジネスの分野で幅広く利用されています。プログラミングの学習もターボパスカルから始めれば、ベーシックで貧弱な言語で学習したために時間を無駄にする、ということはありません。しかもそのターボパスカルが、複雑で難解だったWindowsのプログラミングを簡単にしてくれるのです。



Turbo Pascal for Windowsには、Object Windowsというアプリケーションの骨格になるクラスライブラリが付属しています。ここからウィンドウやメニュー、ダイアログ、コントロールなどを継承すれば、最初からコードを記述する必要はないのです。オブジェクト指向の拡張がされていないC言語でのプログラムより、格段に素早く、短いコードでアプリケーションを作成できるようになります。Turbo Pascal for Windowsは、自分自身がWindowsのアプリケーションであり、Windowsプログラムのみの作成用に設計されています。



ソースコードの編集、コンパイル、実行、デバッグなど、使いやすさで定評のある統合環境を用い、全てWindows上で行えます。ダイアログ、メニュー、ビットマップ、カーソル、アイコン、文字列リソース、アクセレータ、フォントなどのWindowsリソースを視覚的に作成したり、編集することができるリソースワークショップも付属していますから、高額なSDKを別途購入する必要もありません。Turbo Pascal for Windowsには、Windowsアプリケーション作成に必要なツールの全てが、ひとつのパッケージに含まれているのです。

■動作環境：PC-9800シリーズ（ハイレゾモード対応、LTは除く）、PC-286/386シリーズ/MS-DOS Ver 3.1以上/MS-Windows 3.0以上必須/マウス：バスマウス/要ハードディスク

■価格……………………29,800円（税別）

NEW

TURBO PASCAL®
FOR WINDOWS

ユーザ向け情報誌
LANGUAGE EXPRESSプレゼント!!

■この広告をご覧になって、製品に関する資料を請求していただいた方全員に、ボーランド発行の情報誌「LANGUAGE EXPRESS」をさし上げます。

※All Borland Products are trademarks or registered trademarks of Borland International, Inc. ©Borland International, Inc.
※Microsoft、Windowsは米国マイクロソフト社の登録商標です。その他、商品名は一般に各社の商標です。

●製品に関するお問い合せは、総販売元㈱マイクロソフトウェアアソシエイツ TEL.03-3486-1411までお願いします。

株式会社ボーランドジャパン
〒107 東京都港区南青山7-8-1 小田急南青山ビル TEL.03-3486-1400㈹

〈資料請求番号002〉

次代のアドバンテージ、最新の技術をあなたに！

Microsoft® University

独学より大学。

マルチメディアのプラットフォーム MS-Windows、高信頼性の MS OS/2。本格的なマルチタスクの実現、グラフィカルなユーザ・インタフェース(GUI)、LANマネージャによる高度なネットワーク処理など、技術者の創造力を刺激する環境を提供します。マイクロソフト・ユニバーシティは、その登竜門。ウインドウ環境、デバイスドライバの開発、ネットワーク処理など、アプリケーション・ソフト開発に欠かせない、ノウハウをレクチャします。入門から専門まで、充実したコースラインアップ。短時間で即開発に結びつく新しいコンピューティングスタイルを身につけて下さい。

Microsoft Windows Programming Environment
ウインドウズ・プログラミング・エンバイラメント

■期間：5日間 ■受講料：18万円(税別)

■MS Windows Ver3.0のアプリケーションを作成する方のためのコースです。Windows プログラミングのポイントを重点的に解説、実習中心のコースです。

Microsoft Windows Applications Programming
ウインドウズ・アプリケーションズ・プログラミング

■期間：5日間 ■受講料：18万円(税別)

■MS Windows Ver3.0対応のアプリケーションを開発するプログラマや設計者のためのコースです。ユーザインタフェースやアプリケーションなど設計ポイントがマスタできます。

Microsoft LAN Manager Programming Environment
LANマネージャ・プログラミング・エンバイラメント

■期間：4日間 ■受講料：15万円(税別)

■マイクロソフト LAN マネージャ(LM)用アプリケーションを開発する設計者のためのコースです。LMのアーキテクチャと特徴について解説します。

Microsoft LAN Manager Applications Programming
LANマネージャ・アプリケーションズ・プログラミング

■期間：5日間 ■受講料：18万円(税別)

■マイクロソフトLAN マネージャ(LM)上で動作するアプリケーションの作成や移植するプログラマのためのコースです。

MS OS/2 Presentation Manager Programming Environment
プレゼンテーション・マネージャ・プログラミング・エンバイラメント

■期間：5日間 ■受講料：18万円(税別)

■MS OS/2プレゼンテーション・マネージャのアプリケーションを作成するための入門コースです。作成に必要となる基本事項を説明いたします。

MS OS/2 Presentation Manager Applications Programming
プレゼンテーション・マネージャ・アプリケーションズ・プログラミング

■期間：5日間 ■受講料：18万円(税別)

■MS OS/2 PMのアプリケーションを作成するためのステップアップコースです。経験者の方を対象に、より詳しく説明します。

MS OS/2 Programming Environment
プログラミング・エンバイラメント

■期間：4日間 ■受講料：15万円(税別)

■MS OS/2のアプリケーションを作成するために必要となる基本的技術を修得するコースです。本コースは、OS/2に取り組む上での基本コースとなります。

MS OS/2 Applications Programming
アプリケーションズ・プログラミング

■期間：5日間 ■受講料：18万円(税別)

■MS OS/2のAPIを使用し、マルチタスク、メモリ管理、プロセス間通信、ダイナミックリンクなどの機能を有効に活用する、実習を中心としたコースです。

MS OS/2 Device Drivers
デバイス・ドライバ

■期間：4日間 ■受講料：15万円(税別)

■MS OS/2のデバイス・ドライバの機能、構造を修得し、実際にデバイス・ドライバを作成するコースです。デバイスヘルパールーチンなども解説します。

C言語入門

■期間：2日間 ■受講料：6万円(税別)

■これからC言語を始めようとする方を対象に、C言語の基本的な文法からわかりやすく解説します 統合環境を持つQuick Cを使って理解を深めます

C言語応用

■期間：3日間 ■受講料：9万円(税別)

■関数、ポインタ、構造体についてプログラムを作成しながら詳しく説明します セミナは、最新のMicrosoft C Ver 6.0を使用して実施いたします

●お問い合せは

富士ソフトウエア株式会社 MU教育センタ

TEL.03-5461-8296 FAX.03-5461-8469

日本電気株式会社殿の御協力によりPC-9800シリーズ対応コースも好評開催中!!

Microsoft®
マイクロソフト株式会社
〒160 東京都新宿区西新宿7-5-25 Kビルディング

FSI 富士ソフトウエア株式会社
MU教育センタ
〒108 東京都港区港南2-12-27イケダヤ品川ビル

〈資料請求番号003〉

BORLAND
The Leader in Object-Oriented Programming

問：Windows™ アプリケーション開発に最も有効なコンパイラは、何ですか？

（解答はページ下をご覧下さい）

(Inactive C:¥WORK¥BORLAND.EXE)

	従来の開発環境	？
Cコンパイラ	¥ 98,000	¥ 98,000
C++コンパイラ	なし	含む
Windows AP作成ツール	¥ 98,000	含む
アセンブラ	¥ 40,000	含む
合　計	¥236,000	¥ 98,000

※上記の価格には、消費税は含まれていません。

パソコン上に、視覚的で快適な操作環境を提供するWindows 3.0。そこで利用するアプリケーション作成のためのコンパイラは、現在のものでベストといえるでしょうか？機能、スピード、価格…。これらの面で理想的ともいえるコンパイラが、登場しました。「BORLAND C++」です。BORLAND C++は、Windowsアプリケーションを開発、デバックするC/C++コンパイラとツールを標準で装備しています。

■Windowsアプリケーション開発をサポート

BORLAND C++は、Windowsプログラムを完全にサポート。高速かつ高性能な処理を実現しています。MDI(Multiple Document Interface)やDLLs(Dynamic Link Libraries)といったようなWindowsの機能を使って、大規模なプロテクトモードアプリケーションの開発が可能になります。さらに、MicrosoftのリソースコンパイラとWINDOWS.H、Resource Work Shop(日本語版)が標準装備されていますので、別途の開発キット(SDK)は不要です。

■ANSI C／AT&T C++両用コンパイラ

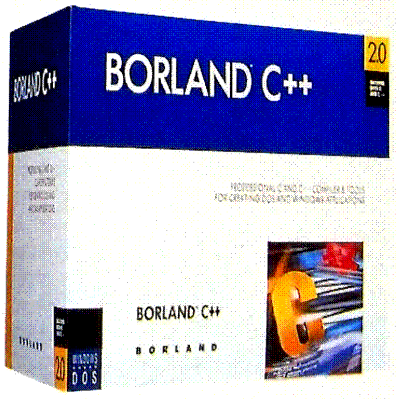
BORLAND C++は、CとC++両方のコンパイラとして利用できます。さらに、再コンパイルの速度を大幅に向上させる「コンパイル済みヘッダ」機能をサポート。MS-DOSの640KBというメモリの制限を破るオーバーレイマネージャVROOMの利用や、Turbo Driveによってプロテクトモードで動作する統合開発環境での大規模なプログラムの作成も可能です。

■Programmer's Platform

Programmer's Platformは、好きなツールを組み込む機能を持つ強力な統合開発環境で、MS-DOSやWindowsアプリケーション開発に必要なすべてのツールがすでに組み込まれています。さらにオーバーラップウインドウやマウス、そしてWindowsのプログラミングリファレンスを含んだオンラインヘルプをサポート。内蔵のマルチファイルエディタは、アンドゥ(実行の取消)／リドゥ(再実行)やマクロ機能を装えています。

■動作環境：PC-9801シリーズ(ハイレゾモード対応、LTは除く)、PC-286/386シリーズ／メモリ：640KB以上／MS-DOS：Ver 3.0以上／MS-Windows 3.0以上／マウス：バスマウス／要ハードディスク

■価格…………………………**¥98,000**(税別)



ユーザ向け情報誌
LANGUAGE EXPRESSプレゼント!!

■この広告をご覧になって、製品に関する資料を請求していただいた方全員に、ボーランド発行の情報誌「LANGUAGE EXPRESS」をさし上げます

(答) BORLAND® C++

* All Borland Products are trademarks or registered trademarks of Borland International, Inc. ©Borland International, Inc.
* Microsoft, Windowsは米国マイクロソフト社の登録商標です。その他、商品名は一般に各社の商標です。

● 製品に関するお問い合せは、総販売元㈱マイクロソフトウェアアソシエイツ TEL.03-3486-1411までお願いします。

株式会社ボーランドジャパン
〒107 東京都港区南青山7-8-1 小田急南青山ビル TEL.03-3486-1400代

〈資料請求番号004〉

Programmer's Page
MONTHLY MAGAZINE FOR PROGRAMMERS

特集
やっぱり僕らはBASIC!
用途を選ばぬそのプログラミングパワーを探る

最先端の仕様を満足する美しいBASIC
データ構造＋プログラム構造＝ニューベーシック　市川新　2

N88BASICからQuickBASICへ
構造化プログラミングを学ぶ　塚本吉彦　11

プロジェクト指向環境を備えた先進のBASIC
Microsoft BASIC Ver.7.1　海老原浩之　16

512倍簡単になるWindowsプログラミング
Microsoft VisualBasic　酒井法雄　25

企業に生きるBASIC　50万ステップにおよぶBASICシステム　加藤雅彦　38

簡易性，多機種対応など教育用に適した特色
BASIC/98　木戸能史　40

生みの親が作った由緒正しい処理系
TrueBASIC　54

プログラム「遊」工房　拾壱乃巻
「JOKE/TOKEI」プログラム　谷口典生　74

IC千夜一夜　第6回
もうひとつの8ビットCPU　ーMC6800ー　虎岩登　80

MS-DOSツール・ラボ　第9回
フリーメモリ残量調整プログラム　REMMEM.EXE/RELMEM.EXE　加野島英渡　84

実践C塾　基礎テクニック篇　第6回
ちょっと便利な関数群　大野元久　94

98活用アセンブラ技法　タイマ編　第2回
時間取得・表示プログラム／プリンタスプーラの作成　星野操　101

付録
5”2HD フロッピーディスク

綴じ込み付録
QuickBASIC Ver.4.5ステートメント・関数一覧

表紙イラスト：日暮修一
表紙AD：佐々木幹夫(SE)

パソコン博物館　第12回
はじめての16ビット機／三菱電機 MULTI-16　井戸禎光 …… 70
プログラマの道具学　第12回
電子メール　祐安重夫 …… 72
情報処理技術者試験　第２種合格！トレーニング
第９回　FORTRAN　稀代智明 …… 113
Book Review「UNIX4.3BSD の設計と実装」　中村正三郎 …… 118
本のアンテナ　古田島義和
SE-NET 拾い読み …… 120
西川和久の希望的観測 …… 122
コンピュータの法律　熊谷信太郎 …… 124
アキバ短信
プログラマの仕事場　第９回　株式会社スタジオ・ゲン …… 126

INFORMATION PAGE …… 129
NEWS & NEW PRODUCTS …… 130
Programmer's Program　Newton-Backup …… 134
オンラインソフト採集帳12「LHQ」 …… 136
HEALTHY? UNHEALTHY?
ドラゴン・ポスト …… 138
付録ディスクのご案内 …… 140
バーチャルスペースガイド …… 141
読者プレゼント
クイズ・ランダムスクリーン …… 142
オンラインサービス（SE-NET）のお知らせ
バックナンバーのご案内 …… 143

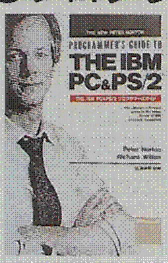
広告索引　　資料請求番号

人材募集広告

1991 DECEMBER No.12

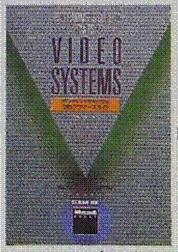
翔泳社が贈るプログラマ必携書

IBM PC & PS/2 プログラミング 3部作



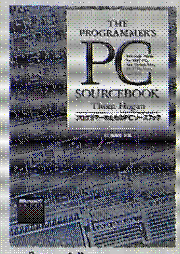
あのピーター・ノートンのベストセラー。IBM プログラマ必携のスタンダードガイダンス。

IBM PC & PS/2
プログラマーズガイド
● Peter Norton & Richard Wilton著
●定価　5,200円



IBM PC ファミリーのビデオシステム完全ガイド。

PC & PS/2 ビデオシステム
プログラマーズガイド
● Richard Wilton著
●定価　6,000円



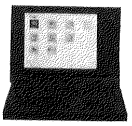
IBM PC & PS/2 プログラマ必携の技術資料宝庫。

プログラマーのための
PC ソースブック
● Thom Hogan著
●定価　5,200円

IBM Authorized Dealer

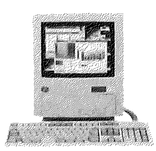
NEWマシン ぞくぞく大量入荷!!

PS/55 note



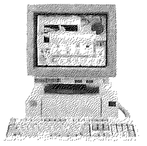
大容量80MBハードディスク内蔵 S18 SVa S28 SA8
パワフルなDOS/Vノート
80386SX(16MHz)2MB FDD内蔵

PS 55Z 30U



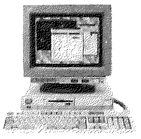
XGA採用パーソナルディスクトップ
80386SX(20MHz)2MB or 6MB
パソコンらしくないパソコン?
オシャレ

PS 55Z 10S



386SX搭載の
スタンダードPC
DOSV、VGA、ATバス採用
80386SX(20MHz)2MB or 6MB

PS 55Z 10T



PS/55Z
最強パフォーマンスPC
XGA高解像度・大容量
80386(20MHz)2MB or 6MB

NEWS とうとう登場!

日本語スーパーVGAドライバー by CFC
噂の800×600ドライバがとうとう出た
今すぐT·ZONEへ行こう!

800×600ドットのSuper VGA(S-VGA)を日本語対応にするスクリーンドライバ。

T·ZONEならでは 大特価 ¥2,500

80486/33MHz made in USA
先着20台限り 超特価¥498,000

T·ZONE

80486/33MHz	
RAM 4MB(max 32MB)	2シリアル、1パラレル
64KBキャッシュ(max 256KB)	拡張スロット×8
3.5" 1.44MB×1/5" 1.2MB×1	ドライブベイ×6
CONNER 120MB HDD	ミニタワー
スーパーVGAアダプタ512KB	アルプス製101キーボード

80386/33MHz ノート made in TAIWAN
限定20台 超特価¥438,000

80386/33MHz	外部VGA出力
RAM 2MB(max 16MB)	1シリアル、1パラレル
32KBキャッシュ	外部テンキー付属
3.5" 1.44MB×1	外部101キーボード、外部FDDポート
40MB HDD	ニッカドバッテリー(充電2時間、2時間使用)
32階調VGAバックライト液晶	280W×220D×54H 4kg

マウス シリアル 超特価¥8,000~ PS/2 超特価¥8,000~

PS/55 NOTE用 2MBメモリー 超特価¥28,000~
4MB 超特価¥48,000~

Microsoft BALLPOINT MOUSE 超特価¥28,000

VGA信号をビデオに変換するコンバーター 超特価¥98,000
320×200~640×480の信号をS-VHS、ビデオに変換して、テレビやビデオに接続できます。

WINDOWS 3.0
時代のトレンドWINDOWS 3.0がIBMではマルチウィンドウズで複数のDOSアプリが同時実行可能です。

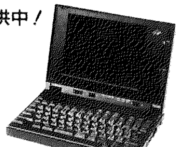
IBM 日本語WINDOWS 3.0	¥20,000
ADDBE TYPE MANAGER	¥20,000
AFTER DARK	¥9,800
BECKERTOOLS	¥16,000
MICROSOFT ENTERTAINMENT PACK	¥8,800

その他WINDOWS用アプリケーションが200種類以上あります。
WINDOWS 3.0対応のPDS(無料)もあります。

NOTE
ノートといえば、PS/55 note 大特価提供中!

IBM	PS/5523SJ1	~~¥248,000~~
IBM	PS/5523SJ4	~~¥368,000~~
SANYO	MBC-18NVH4	~~¥355,000~~
NBCC	386NOTE	¥298,000
NBCC	386LAPTOP	¥266,000
NBCC	286NOTE	¥244,000
NBCC	286LAPTOP	¥220,000

PS/55 note



DOS/V
DOS/VといえばT·ZONEです。入門機から互換機まで!

IBM	PS/5530-SW4	大特価提供中! ~~¥608,000~~
	標準で6MBのRAMと日本語WINDOWS 3.0が付属しています。別売DOS/Vも付いています。	
IBM	PS/5510ZJ2	大特価提供中! ~~¥214,000~~
IBM	PS/5510ZJ4	大特価提供中! ~~¥398,000~~
IBM	PS/5510TJ4	大特価提供中! ~~¥564,000~~
ALR	POWER FLEX 20CSX, RAM 5MB, 3.5" 85", 40MBHD	¥385,000
ALR	BUSINESS VEISA 486/33, RAM, 2MB, 3.5" 85", 120MBHD	~~¥1,250,000~~
東芝	DynabookでDOS/Vが動いています(2Fでデモ中)	

VGA
DOS/Vの必須アイテムはVGAアダプターです。
豊富な在庫から選べます。

BASIC VGA 640×480	¥19,800
SUPER-VGA 1024×768 16色	¥38,000
SUPER-VGA 1024×768 256色	¥44,800
Dyanbook用VGAアダプタ	¥49,800

VGAモニター
DOS/V対応やWINDOWS用として高解像大画面のモニターがほしいあなた、何を選びますか?

NANAO	T660i (ケーブル別)	20" 0.31mm 30~78kHz	1284×1024	~~¥598,000~~
三菱	XC-98V3	14" 0.28mm 98/VGA	640×460	¥78,000
ACER	7015	14" 0.31mm 15.0-36kHz	800×600	¥85,000
日立	CM1483MU	14" 0.28mm 30-40kHz	1024×768	¥85,000
NEC	MULTISYNC II	14" 0.31mm 15.5-35kHz	800×560	¥95,000
NEC	MULTISYNC 3D	14" 0.28mm 15.7-38kHz	1024×768	¥180,000
NEC	MULTISYNC 4D	16" 0.28mm 30.0-57kHz	1024×768	¥280,000
IBM	8515	14" 0.28mm VGA, 8514/A, XGA		~~¥160,000~~

SCSI I/F 超特価¥9,800~ SCSI I/F, W/FDC 超特価¥15,000~

MIMI-SCSI 超特価¥29,800
プリンタポートにつなくだけでSCSIデバイスが使えるようになります。
87MB HDDセット 超特価¥138,000

Mitac

3060D	101キーセット	106キーセット
RAM 3MB、HDD 40MB	¥318,000	¥333,000
RAM 3MB、HDD 120MB	¥368,000	¥383,000
RAM 5MB、HDD 120MB	¥385,000	¥400,000

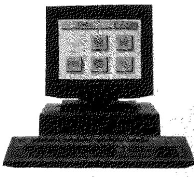
共通仕様
80386SX 16MHz、3.5" 1.44MB×1、5" 1.2MB×1、スーパーVGA(512KB)、スーパーVGAモニター

GAME
時代はゲーム!? TERADRIVEや新作のソフトが山盛りでご来店をお待ちいたしております。

SEGA TERADRIVE MODEL1	~~¥148,000~~ 大特価提供中!
SEGA TERADRIVE MODEL2	~~¥188,000~~ 大特価提供中!
SEGA TERADRIVE MODEL3	~~¥248,000~~ 大特価提供中!
SEGA TERADRIVE 専用モニター	~~¥89,800~~ 大特価提供中!
SEGA TERADRIVE マウス	~~¥6,800~~ 大特価提供中!

WALT DISNEYの永遠のアイドル、ミッキーマウスがあなたにしゃべりかけます。

MICKEY'S ABC	¥10,800
MICKEY'S 123	¥10,800
MICKEY'S ABC & SOUND SOURCE	¥13,800
SOUND SOURCE(驚異のローコスト高級スピーカーです)	¥6,800
ADLIB FOR ATバス	¥30,000
ADLIB FOR MCAバス	¥40,000
SOUND BLASTER FOR ATバス	¥29,800
SOUND BLASTER FOR MCAバス	¥48,000
JOYSTICK	¥4,800
ゲームカード(PC/AT)	¥9,800
ゲームカード(MCA)	¥15,000



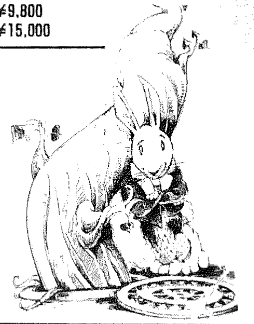
the Manhole (日本語版)

それはマンホールからはじまった……
ゲームを超えた
ファンタジーアドベンチャー発売中!

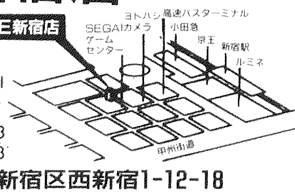
DOS/V対応
IBM PS/55
テラドライブ対応

DOS/V OADG

¥9,800



T·ZONE 新宿店

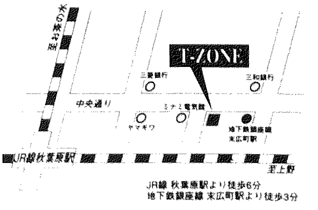


1F APPLE & NEC
03-3342-1901
2F IBM PS/FACTORY
03-3342-5533
FAX 03-3342-1903
〒160 東京都新宿区西新宿1-12-18

T·ZONE

営業時間
AM10:30~PM7:00

PS PLAZA
IBMゾーン
3F ハードウェア
4F ソフトウェア
☎03-3257-2652
FAX 03-3257-2661



〒101 千代田区外神田4-4-1

T·ZONEスタッフ募集 IBM、Apple、無線のスタッフを募集中です。アルバイトも募集中! ☎03-3257-2690

☆T·ZONEの7Fは専門書籍(国内・外)・データーブックの専門フロアです。大量在庫中! ほしい本は何でも揃います。

<資料請求番号005>

特集

やっぱり僕らはBASIC!

用途を選ばぬそのプログラミングパワーを探る

BASICは，1964年にプログラミングを教えるための言語として開発された．マイクロコンピュータが登場するとすぐさまBASICは移植され，パソコン黎明期におけるアプリケーション開発言語として活躍することになる．1982年には，日本においても8ビットパソコンを中心に未曾有のBASICブームを巻き起こしている．しかし，構造化プログラミングの提唱とともに「BASICでプログラムを覚えた人間は一生まともなプログラムが書けない」などといわれるようになり，現在では主要なプログラミング言語としての地位をC言語に譲り渡してしまった．

一方で，BASICはその後も進化を続け，現在では行番号を廃止し，構造化ステートメントを取り入れたモダンな高級言語に生まれ変わっている．この新しい“Beginner's All-purpose Symbolic Instruction Code”の実力を探ってみよう．

Beginner's
All-purpose
Symbolic
Instruction
Code

特集……やっぱり僕らはBASIC！

最先端の仕様を満足する美しいBASIC

データ構造＋プログラム構造＝ニューベーシック

流通経済大学経済学部助教授
市川　新

Beginner's
All-purpose
Symbolic
Instruction
Code

わが国では，1994年度までにすべての小，中，高校にパソコンが導入されることになっているが，ここで標準コンピュータ言語にBASICが採用される可能性が高い．こうなれば，BASICは，外国語としての英語に匹敵するものになると思われる．教科書で採用されるBASICは標準のものであることが望ましいが，日本工業規格で1982年に定められたJIS BASICは「基本BASIC」と呼ばれる古い規格のものだ．しかし，現在パソコンで使われている標準BASICは「業界」標準であるマイクロソフトBASICである．ここでは，BASICの発展と進化の歴史をふり返るとともに，進化し，洗練されているニューベーシックの機能と今後の標準BASICについて考えてみたい．

読み・書き・ベーシック

手元にある最新の高等学校数学科用「数学II」[1]を開いてみよう．1章は「いろいろな関数」で34頁，2章は「数列」で20頁，3章は「微分とその応用」で26頁，4章は「積分とその応用」で20頁，5章は「ベクトル」で24頁，6章は「確率と統計」で48頁，そして7章は「コンピュータの利用」で20頁である．わが国の情報教育は，その情報機器生産能力に比べて，先進国で極めて遅れているといわれる．それでも，ようやく，高校の普通科の教科書の中で取り上げられるようになった．コンピュータについて，すでに，工業高校や商業高校卒業者を対象として，ごく少数の大学で入学試験に出題されている．これからは，より一般的に入試問題として出題され，塾の補習科目になるかもしれない．

「コンピュータの利用」の教材は，パソコンベースのコンピュータ言語BASICである．学校教育のベーシックについて再現してみよう．

①直接モード

ベーシックにおける直接モードとは，PRINT文によりその引数の式を直接に評価をさせるものである．もちろん，変数を含む式をあらかじめ評価しておくこともできる．しかし，入力したプログラム(厳密には実行順序を表現していないのでプログラムにならない)を保存し，再編集することはできない．直接モードの利用は電卓的利用といえる．

②間接モード

一連の命令文を記述し，その実行順序をあらかじめ指示しておけば，プログラムといえる．個々の命令文にはユニークな行番号を付加することによって，行番号の若い順に実行が自動進行する．自動進行を開始する指示はRUN文である．

③変数

プログラムに変数を使うことができる．直接モードは電卓的利用といえるが，電卓では変数を使うことができない．ベーシックでは変数を使うことができるが，しかし，その変数が参照されるときまでに，変数に値をもたせておくことが要求される．

④命令

PRINT文と代入文で代表される．この教科書では，代入文中のLET語が省略されている．代入文の「＝」には左辺と右辺が同値であるという意味はない．懐かしいあの♪ Let it be ♪の「be」の意味であるから，LET語は省略しないほうが望ましい．

⑤関数

平方根，三角関数，指数関数，対数関数があげられる．数学Iと数学IIの範囲からすれば当然であるが，三角関数では尺度としてラジアンが使われることが指摘されている．

⑥実行制御

GOTO文，IF～THEN文，FOR／NEXT文，および，END文である．つまり，コンピュータプログラムに必要とされる制御構造が一通り取り扱われる．

⑦副プログラム

GOSUB／RETURN文によるサブルーチンの利用がプログラムの分割の方法として紹介される．

⑧編集

プログラムの編集は，行番号の利用，LIST文，NEW文，および，制御キーの利用が採用されている．

⑨例題

5行から20行程度のプログラムで，素数の知識を前提とする例題が取りあげられている．

⑩アルゴリズム

2次方程式の根の判別による分岐を含めたフローチャートがアルゴリズムの表現にされている．

ベーシックプログラミングの経験者であれば，今，高等学校で実習されているプログラミングの内容が想像できるであろう．義務教育の中学校では92年度から「コンピュータ教育」が始まる．さらに，小学校へと「第三の波」は押し寄せる．そうすれば，日本の社会の隅々までに情報社会は浸透したことになる．

ところで，わが国の学校教育にパソコンが制度的に導入されたのは85年度に遡る．国費補助により，94年度までにすべての小，中，高校にパソコンが導入される．それにともない，教員に要求される経験と知識に，当然のことながら，「コンピュータの経験と知識」が含まれてくる．90年度以後に大学などに入学し教員免許を取得しようとする学生は教育技術に関する科目の取得が要求されるようになった．この内容は基本的に情報教育に関わるものでコンピュータ利用の経験が含まれている．いいかえれば，94年以後の新任の教員は，その所持する免許の教科に関わらず「コンピュータの経験と知識」が期待される．

このように，制度の改革とハードウェアの導入の矢は放たれた．84年当時，欧米とくに米国ではアップル社の攻勢もあり，小学校でも積極的なパソコン導入[2]が行われていた．すでにロゴ語(Logo)が教材としてかなり採用

されていた．ただ，米国の場合，その内容の質と設備の量的問題は学校の所在地の経済力に左右され，日米比較のための評価基準として平均値は役に立たない．それだけ，豊かな地域と貧しい地域の格差が激しいことになる．それに比べて，同質であることがわが国の基本政策であり，学校教育での情報教育の導入が先進工業国中で最後尾であるのは，関連する技術が成熟するまで待っていた結果といえよう．小中学校でのパソコンの使われ方が実際にどうなるかは関係者の努力しだいになろう．どのようなソフトウェアを用意し利用することになるのだろうか．

しかし，パソコンのソフトウェアというと連想するのはやはりベーシックである．ご存知のように，ハードウェアの基本ソフトウェアとして必ず装備されているのがベーシックである．また，法的にも後ろめたさがなく使えるのはベーシックである．おそらく，コンピュータ言語としてのベーシック語は，外国語としての英語に成長することになろう．

標準ベーシック語 VS JISベーシック語

教科書で採用されているベーシック語は本来標準語であることが望ましい．当然のことながら，前述の数学IIのベーシック語は標準ベーシック語のはずである．そこで，日本工業規格を調べてみるとJISベーシック語が制定されている．1982年に規格制定されたもので「基本BASIC」と呼ばれている．将来の「拡張BASIC」の規格化を想定して名称が定められたのであろう．

このJISベーシック語に準拠したベーシック語にお目にかかることはほとんどない．つい最近までは，いわゆるメインフレームコンピュータにJISベーシック語が提供されていた．しかし，標準ベーシック語に慣れた利用者がこのベーシック語を利用すると，まるで別の新しい言語のように思われるらしい．同一のベーシック語に見えるのであるが移植性は極めて悪い．結局，誰も使わないベーシック語になるので，メインフレームメーカーも技術的改善を行わない．どうしてもJISベーシック語を使うのであれば，メーカーに依頼することになる．SEがJISベーシックの磁気テープを持参するようであるが，そのバージョンは恐ろしく古い．

つまり，標準ベーシック語として知られているベーシック語とJISベーシック語とは基本構文こそ共通であるが，使われる語彙とその意味にはほとんど共通性はない．正確にいえば，標準ベーシック語の語彙の豊富さとその意味の強力さがJISベーシック語にないのである．利用者の計算資源の利用目的は多様化しており，それにともない期待する処理機能も多様化している．標準ベーシック語の特徴は，ハードウェアに対応した機能の多様性にあるといってよい．

学校教育英語が英国英語から米国英語に変わってしまった．JISベーシックは元祖ベーシックに準拠し，標準ベーシック語は「業界」標準であるマイクロソフトベーシック語そのものである．JISベーシック語をさておいて，一私企業の製品が教科書にとりあげられている例は，マイクロソフトベーシック語だけであろう．トロンプロジェクトを核としたパソコンOSの標準化をめぐり，一時，内外各メーカーの営業政策がぶつかりあった．やはり，標準パソコンOSは棚上げにされ，「業界」標準パソコンOSの採用で落ち着いたようである．

バック・トゥ・フューチャ

現在多くのコンピュータ言語が使われているが，ベーシック語にはこれら社会的背景のみならずコンピュータの発展と同期して進化してきた独自の歴史がある．歴史をふり返ることは，この言語の進化の方向を予測することと同じ意味をもつ．JISベーシック語は，わすれられた言語のように思われるかもしれないが，元祖ベーシック語の血統書が付いた直系のベーシック語である．

ダートマス大学のケメニー教授とカーツ教授は，40KBの主記憶容量のコンピュータを20ユーザーが同時に利用できるタイムシェアリングシステムを実用化したことで知られている．さらに彼らが学生の入門用に開発した最初のベーシックは，1964年5月1日午前4時[3]に最初のメッセージを打ち出したといわれる．バッチ処理とは異なり，タイムシェアリングシステムでは対話処理となり，それまでのバッチ処理とは異なる新しいプログラミング環境が求められる．現代の標準ベーシック語はインタプリタ方式であるが，最初のベーシック語はコンパイラ方式であった．

プログラムの記述には，当時のマンマシンインターフェースであるテレタイプ型端末を利用するから，ラインエディタを必要とする．エディタで記述されたプログラムは，コンパイラにより機械語に翻訳される．機械語のプログラムを実行すれば，ベーシック語で表現された処理が再現される．これら

BASICプログラミング用語集
ISO　国際標準化機構（International Organization for Standardization）の略．国際的な規格の開発と標準化を促進する機関．

の一連の操作は，対話的に進行することがコンピュータについて特別な知識をもたない利用者には望ましい．タイムシェアリングシステムによって実現された対話環境とベーシック語は，その後の10年間に主要汎用コンピュータメーカーにより，独自の仕様で開発され，ベーシック語方言の豊富さにつながる．

マイクロプロセッサが出現すると，ベーシックの移植が行われるようになる．1974年は，インテルの8008用にベーシックが開発された．さらに，8080を実装したマイコンが商品化されるとやはりベーシックが開発された．今でもタイニーベーシックとして記憶に新しい．もちろん，数Kバイトのメモリ容量ではその機能は極めて限られたものになる．

しかし，ベーシックが広く社会定着し，日本の教科書にそれが出現することになった貢献者は，W.ゲイツ氏とP.アレン氏の開発したベーシックであろう．彼らは標準ベーシック語の生みの親といってもよい．この商業的成功により，ご存知のマイクロソフト社の設立につながることになる．元祖ベーシックを先頭に1980年までに多種多様のベーシックが開発され，それぞれが必要に応じて置かれた環境に適応するために進化を遂げたのであるが，基本構文こそ元祖ベーシックに共通であるものの，相互に交流できないくらい方言化した．

このようなある種の混乱の原因は，ケメニー教授とカーツ教授がベーシックに反映させた理念が広く支持されたことの裏返しではなかろうか．第1に，利用者指向であること，それも最終利用者がその時点で獲得している経験と知識に対して，必要最低限のコンピュータ言語の知識でコンピュータを道具化できることを目差した．第2に，オペレーティングシステムの機能を含むプログラミング環境を具体化したことがあげられる．現在，標準ベーシックであれば，一部の入出力系の命令を除けば，パソコンであれば機種依存する部分は少ない．日本のパソコンメーカーはハードウェアレベルでの自社製品の違いを強調しているにもかかわらず，標準ベーシックの環境にあれば利用者はその違いにほとんど煩わされることはない．したがって，学校がどのメーカーのパソコンを導入しても操作性で問題になることはない．教科書に標準ベーシックを教材として採用できることになる．

混沌 VS 進化

米国におけるベーシックの規格化は1978年に遡る．この米国の「基本BASIC」の規格はケメニー教授とカーツ教授のベーシック1965年版に基づいている．JISの「基本BASIC」はこの米国版に準拠したものである．前述したように，メインフレーム系のベーシックは，これら「基本BASIC」に基づいてそれぞれのメーカーが＋αの機能の拡張を行ったものになる．この意味では，元祖ベーシックの面影を残している．

しかし，元祖ベーシックは，その理念を発展させ，変数の命名に複数文字を許し，制御構造を拡張し，外部サブルーチンの概念を取り込み，コンピュータ言語としての進化をはじめていた．

また，マイコンの高性能化や半導体素子の低価格化，また，マンマシンインターフェース装置の多様化にともない，それら実装されたハードウェアを利用者が応用可能な機能をベーシック語に吸収したために，より高度なプログラミングが可能になりはじめた．さらに，初期のメモリ容量による制約が解決されはじめた．とはいえ，限られたハードウェア資源の中で高水準コンピュータ言語風に見せ，最低限のプログラミング環境を付加した標準ベーシックの生い立ちは，変数管理や制御構造を洗練させる余裕がなかったと思われる．結果的に，美しいコンピュータ言語を求める専門家からの「教育的ではない」という批判に甘んじることになる．実際，まとまったプログラムを記述するには，相当の経験と知識が必要[4]である．標準ベーシックに真面目に取り組むと「ビキナー向け」ではなく，アセンブラ語の構造化プログラミングの経験がものをいうことになり，「プロフェッショナル向け」の“PASIC”ではないかと思われる．

横道にそれるが，いま流行のシー語（C言語）でプログラムを記述すると，ベーシック語とシー語の相互関係に気づかずにはいられない．それは，一言でいえば，ベーシック語は非コンピュータ専門家向けであり，シー語はコンピュータ専門家向けである．シー語は構造化高水準アセンブラ語であって，その有用性は，個々の具体的ハードウェアなどの違いを吸収しながら，フォンノイマン型コンピュータのハードウェアの原理を陽に言語に反映させたものである．ベーシック語はグローバル変数の言語であり，シー語はローカル変数の言語である．したがって，シー語の得意とする分野はハードウェアの知識が要求される．それに対して，ベーシック語は，コンピュータシステムの違いを吸収し，ハードウェアの知識を必要としない．したがって，オペレーティングシステムの知識も必要とし

BASICプログラミング用語集
IEEE　アメリカ電気電子技術者会（The Institute of Electrical and Electronics Engineers）の略．電気・電子などの関連分野で標準化を進める団体．

ない．ベーシック語を必要とする人は，その人がある特定の分野について経験と知識をもち，ベーシックを使うことによって，今まで以上にその特定の分野の経験と知識が発展させられる人といえよう．そういう「プロフェッショナル向け」である．道具を必要とする目的がなければプログラミングはできないのである．

マイコンの出現によって，そのベーシック語は誕生時にプログラムされていたはずの進化の遺伝子情報を失い，退化したといえよう．それは中水準アセンブラ語と呼んでも差し支えないであろう．しかし，マイクロプロセッサの進化とともに，再び進化を開始するのであるが，1度退化後のUターン型進化の系統樹は独自の形を造る．

マイコンがパーソナルコンピュータに進化すると，メインフレームコンピュータの多数の利用者の1人として利用することと遜色がなくなってきた．パソコンとメインフレームとの機能の差がなくなれば，理想の仕様を実現する標準化に障害になるものはない．1984年に，基本ベーシックの制約を取り払った，ベーシック本来の標準規格案が草案された．ベーシック誕生以来24年の紆余曲折を経て，1988年に米国規格に正式に採用された．

日本では，概ね5年程度の遅れでJIS規格になるようであるから，1993年前後に「拡張BASIC」として出現するであろう．

進化の原点：数値と文字列

ベーシック語の利用者はある仕事をコンピュータに代替させたいという欲求がある．そこで使われるデータは，データ処理の作用の差異から，数値と文字列に分けられる．定量データと定性データといいかえてもよい．筆者は数値データと非数値データとはいわないことにしている．なぜなら，コンピュータのハードウェアと関係のない世界では，まず，定性データがあり，次に定量データがある．そうすると文字列と非文字列（数値）といえなくもない．

ベーシック語が，非コンピュータ専門家向けといえる原点がここにあるのではないかと思うことがある．現在，流通しているベーシックには文字列データ用領域の割り当てとそれに適応した文字列演算機構が論理的に実現されている．もちろん，数値演算機構については物理的に実現されている．広く使われているコンピュータ言語で数値と文字列を同じレベルで認識し実装しているのはベーシックだけである．そのためにシステムプログラムを記述できる．この文字列演算機構は，人工知能向言語であるリスプ語に採用されている処理機構と同じ原理が採用されている．仮想的に無限回数の文字列生成を行うことができる．ベーシックは数値計算専用ではないのである．

数値にしても，非コンピュータ専門家はそれが整数であるか実数であるかとくに意識しない．まして，実数が単精度であるか倍精度であるかは知るはずもない．標準ベーシックではこれらをいつのまにか厳密に区別するようになっていて，アセンブラ語の水準に戻っている．本来，この種の区別ないし知識を必要としないのがベーシックである．

もう1つ，変数宣言を積極的にしないという特徴がある．言語学者の指摘[5]にパスカル語（Pascal）派が聞くと寒気をもよおすものがある．それによると，「普通の人間」には，使用する変数を，あらかじめ，プログラム冒頭などに陽に宣言することは苦痛なことであるということだ．

ベーシックでは，このように，既存の知識体系との親和性に配慮されているといえよう．

嫌われの原点：行番号

ベーシック語は，対話的コンピュータ利用環境における最初の言語である．プログラムの記述もコンピュータを利用して行われる．テレタイプ型端末でプログラムの編集をするには，ラインエディタが便利である．パンチカードをプログラム媒体にすると，やはり，行番号があると整理の都合がよい．

ベーシックが嫌われるのは，この行番号の扱い方に原因があったのではな

リスト1

```
PROGRAM New_basic
  PRINT "Say Hello to the New Basic"
END
```


BASICプログラミング用語集
アイドリング　遊休時間ともいい，CPUが動作可能にもかかわらず動作していない時間のこと．プリンタなど周辺機器の動作時にアイドリングは多い．

リスト２

```
1000 LET X1$="Hello "
1010 LET X2$="the New Basic "
1020 GOSUB 1060
1030 GOSUB 1080
1040 PRINT "Say "; X1$; "to "; X2$; "!"
1050 END
1060 LET X1$="Goodbye "
1070 RETURN
1080 LET X2$="Old Basics "
1090 RETRUN

PROGRAM Internal_Subprgram
  LET X1$ = "Hello "
  LET X2$ = "the New Basic "
  CALL Msg1
  CALL Msg2
  PRINT "Say "; X1$; "to "; X2$; "!"
  STOP

  SUB Msg1
    LET X1$ = "Goodbye "
  END SUB

  SUB Msg2
    LET X2$ = "Old Basics "
  END SUB
END
```

リスト３

```
PROGRAM External_Subprgram
  LET X1$ = "Hello "
  LET X2$ = "the New Basic "
  CALL Msg1
  CALL Msg2
  PRINT "Say "; X1$; "to "; X2$; "!"
END

SUB Msg1
  LET X1$ = "Goodbye "
END SUB

SUB Msg2
  LET X2$ = "Old Basics "
END SUB
```

いか．行番号が編集のための識別子的役割に使われているかぎり問題は生じない．しかし，プログラムの制御の流れで，分岐先を示す識別子の役割と兼用されている．「GOTO　行番号」あるいは「GOSUB　行番号」があると，これらの文で指定された行番号は制御を受け渡される識別子になる．もし，行番号を変更するときは，命令文の引き数の行番号も同時に更新しなければならない．そうすると，ベーシックには専用のエディタが必要となり，利用者が頻度的時間的にもっとも多用する道具であるエディタを選べない．また，必ずしも専用エディタが優秀な製品とは限らない．この仕様の不都合[6]はケメニー教授らも気づいていた．

他のコンピュータ言語をエディタでプログラム記述した経験者は，ベーシック専用エディタを嫌悪するようである．進化したベーシックであるかどうかの簡単でかつ正確な判断基準は，行番号が省略できるかどうかを確かめればよい．行番号が不必要になるとスクリーンエディタが必需品になる．実際，ニューベーシックといえる製品は各社とも行番号が省略できる．

そこで，ニューベーシックの最新版である米国ベーシック規格1988年版[7)8)9)]を概観することにする．以下とくに断らない限り，行番号なしのプログラム例は1988年版に準拠したニューベーシック語であり，行番号ありの例は標準ベーシック語である．さらにニューベーシックの例では，進化した部分を強調することにする．リスト１のプログラムのPROGRAM行は省略可能であるが，本稿の目的から明示する．もちろん，LET文は省略しない．逆にニューベーシックに含まれているGOTO文などの行番号を参照する命令は取り上げない．

リスト１の単位を主プログラムと呼ぶことにする．

先進性の証Ｉ：変数スコープの管理

教科書的例題は別として，実際にプログラムを記述しはじめるとサブルーチンを上手に多用することが定石である．

リスト２に示したプログラムは両方とも，RUNすると"Say Goodbye to Old Basics！"が，表示されるであろう．もともと，標準ベーシックでは変数のスコープの概念がないのですべての変数はグローバル型である．ニューベーシックでは，サブプログラムにローカル変数を定義できる．

リスト３をRUNすると"Say Hello to the New BASIC！"と表示されるであろう．LET文で確保された各文字変数はそれぞれ独立した変数であり，それぞれの主プログラムまたは副プログラムに閉じ込められる．主プログラムのEND文がグローバル変数の有効域を定義する．

リスト 4

```
MODULE Data_and_Process_Abstraction

  PUBLIC NUMERIC N
  PUBLIC STRING S$
  PUBLIC SUB Push, Pop

  SHARE NUMERIC stack(100), ptr

  LET N = 0
  LET S$ = ""
  LET ptr = 0
  EXTERNAL SUB Push(in_data)
    LET ptr = ptr + 1
    LET stack(prt) = in_data
  END SUB

  EXTERNAL SUB Pop(out_data)
    LET out_data = stack(ptr)
    LET ptr = ptr - 1
  END SUB

END MODULE
```

リスト 5

```
1000 LET YN$="Y"
1010 WHILE YN$="Y"
1020 FOR I=1 TO 2 STEP 1
1030 INPUT N(I)
1040 NEXT I
1050 IF N(1)>N(2) THEN PRINT N(1) ELSE PRINT N(2)
1060 INPUT YN$
1070 WEND
1080 END
```

先進性の証II:抽象化技術とオブジェクト指向設計

ソフトウェア開発における最新の話題はデータと手続きの抽象化であり，オブジェクト指向設計である．これを具現化する概念はモジュールの導入である．米国ベーシック規格1989年追補版では，モジュラ・ツー語（Modula2）に採用されたモジュールと同等の機能と表現が追加された．これにより，データの抽象化や手続きの隠蔽が可能になった．

リスト 4 に示すモジュールでは，主プログラムの起動時に 1 度だけ最初に 3 行の LET 文が実行される．それらの中で，PUBLIC 文で宣言された変数がモジュール外から参照できる．SHARE文で宣言された変数はこのモジュール内でしか参照できない．PUBLIC 文で宣言された副プログラムはモジュール外から参照できる．この例にはないが SHARE 文で宣言された副プログラムは，内部で参照できる．SHARE 文によって変数と手続きを隠蔽することができる．

モジュールは，副プログラムとは異なり，その独立性が十分に配慮されていればソフトウェアの保守性が向上できる．独立したモジュールは外部の影響を受けず，機能の信頼性が保証される．グローバル変数とローカル変数の導入，モジュールの実現により，ニューベーシックはコンピュータ言語としても最新の仕様をもつといえる．

先進性の証III:ダイクストラの制御構造

構造的プログラムの基本制御構造は，連続型制御構造，選択型制御構造，反復型制御構造の 3 構造である．これらに対応した表現が豊富であれば構造的プログラミングがたしかにしやすい．しかし，あるプログラムを記述するときに，最初から構造的にあるいはトップダウンに，制御構造を段階的に詳細化できるかというと厳しいものがある．というよりも，ある程度まで試行錯誤して見通しがついたときに，あらためて，構造化的に考えなおすのが，実際

リスト 6

```
PROGARM Mini_Soroban1

INPUT N1, O$, N2

IF O$ = "+" THEN
  LET N3 = N1 + N2
  PRINT N1, O$, N2, "=" N3
ELSE
  PRINT O$, " is a wrong operator"
END IF

END
```


BASIC プログラミング用語集
アルゴリズム　問題から解を得るために定義された明確な手順の集まり．同じ解を得るには複数の手法があり，これによって実行速度や必要な記憶容量が大きく影響される．

のプログラミングと思われる．そうすると試行錯誤の段階でこれらの構造が貧弱であると，プログラミングの効率が悪くなる．ベーシックは，高水準コンピュータ言語の中で，命令の機能の水準が高いこと，命令の種類が豊富であることで，プロトタイピングがしやすいという経験がある．しかし，標準ベーシックではこれら制御構造はリスト5に示す3つしかないところが苦しい．

これに対して，ニューベーシックの基本構造はパスカル語やシー語と比較しても遜色がない（リスト6）．

従来のIF文に新しくブロック型が加わった．ブロック型は連続型命令群を選択する場合に強力であり，視覚的に論理が明解になる．ブロック型にはリスト7の例のように基本命令として多重IF文が含まれる．

選択構造には条件複数分岐型があり，一般にCASE文として各言語に採用されている．ニューベーシックではシー語よりも条件文に融通がきく．

リスト8の例で，後者のSELECT文中のCASE条件は範囲指定をしている．プログラムの内容は，計算結果が，負の1000より小さい場合は2個の負符号を，正の数には正符号を，1000より大きければもう1個の正符号を表示している．

リスト7

```
PROGARM Mini_Soroban2

INPUT N1, O$, N2

IF O$ = "+" THEN
  LET N3 = N1 + N2
  PRINT N1, O$, N2, "=", N3
ELSEIF O$ = "-" THEN
  LET N3 = N1 - N2
  PRINT N1, O$, N2, "=", N3
ELSE
  PRINT O$, " is a wrong operator"
END IF

END
```

さて，最後に反復型制御構造を使ってみる．初心者が反復命令を使いこなせれば，学習者として中級になろう．ニューベーシックではかなり自由度のある構文になっている．完全なプログラム例にすると視覚的に理解しにくいので，反復構造だけを取り出して残りの部分は省略する（リスト9）．

リスト9の両方とも反復からの強制脱出例である．後者は無限反復における打ち切り例である．ところが，反復されているブロックの途中から後半を一時的に無視し，反復を強制続行する命令がない．選択型構造を使うことになるだろうが，CONTINUE FOR文やCONTINUE DO文があってもよいのではないかと思う．反復型制御構造の一般的用法はリスト10のとおりである．

前者の2例は反復回数が0回以上で，後者の2例は1回以上である．反復の条件は，WHILEの条件が真の場合で，UNTILの条件が偽の場合である．

ニューベーシックから標準ベーシックへ

今までの標準ベーシック語はパソコンベーシック語として広く親しまれている．その一方で，コンピュータ言語として洗練されていないという批判もある．標準規格としてのベーシック語は，以上紹介してきたとおり，高水準コンピュータ言語としての最先端の仕様が満足されている．

とくに触れなかったが，行列が基本データ構造に含まれており，行列演算機構も内蔵されている．入出力系につ

リスト8

```
PROGRAM Mini_Soroban3

INPUT N1, O$, N2

SELECT CASE O$
  CASE "+"
    LET N3 = N1 + N2
  CASE "-"
    LET N3 = N1 - N2
  CASE "*"
    LET N3 = N1 * N2
  CASE "/"
    LET N3 = N1 / N2
  CASE ELSE
    PRINT O$, " is a wrong operator"
    STOP
END SELECT

SELECT CASE N3
  CASE IS < 1000
    LET S$ = "-"
  CASE IS < 0
    LET S$ = ""
  CASE 0 TO 500, 500 TO 1000
    LET S$ = "+"
  CASE ELSE
    LET S$ = "++"
END SELECT

PRINT N1, O$, N2, "=", S$; N3

END
```

BASICプログラミング用語集
ANSI　アメリカ規格協会（American National Standards Institute）の略．米国で規格の統一を目的とした団体．日本におけるJISにあたる．

リスト9

```
REM Loop breaking 1
LET J = 1
FOR I = 1 TO 100
  LET J = J * I
  IF J > 100 THEN EXIT FOR
NEXT I

REM Loop breaking 2
LET J = 1
DO
  LET J = J + J
  IF J > 100 THEN EXIT DO
LOOP
```

リスト10

```
REM Loop 1
LET CNT = 0
LET SUM = 0
DO WHILE SUM <= 10000
  LET CNT = CNT + 1
  LET SUM = CNT * CNT
  PRINT CNT, SUM
LOOP

REM Loop 2
LET CNT = 1
LET SUM = 1
DO UNTIL SUM > 10000
  PRINT CNT, SUM
  LET CNT = CNT + 1
  LET SUM = CNT * CNT
LOOP

REM Loop 3
LET CNT = 1
LET SUM = 1
DO
  PRINT CNT, SUM
  LET CNT = CNT + 1
  LET SUM = CNT * CNT
LOOP WHILE SUM <= 10000

REM Loop 4
LET CNT = 1
LET SUM = 1
DO
  PRINT CNT, SUM
  LET CNT = CNT + 1
  LET SUM = CNT * CNT
LOOP UNTIL SUM > 10000
```

いては他の言語に負けない仕様になっている．とくに，対話入力系についてはかなりの小回りが可能である．

ドス系(DOS)で日本語対応版ニューベーシックとして入手できるのは，BASIC/98系，QuickBASIC系，TrueBASIC系の3系列である．それぞれは従来の標準ベーシックより進化している．共通する進化の方向は，行番号を省略できること，グローバル変数とローカル変数の使いわけができることであろう．さらに構造化プログラミングに求められる制御構造対応命令の導入があげられよう．このうち，対応機種が限定されているが標準規格との整合を強調したのがTrueBASIC系，生産性を向上させるプログラミング環境の装備を強調し，ドスメーカーとして他のコンピュータ言語との親和性があるQuickBASIC系，国産異機種のハードウェアの壁を越えて標準ベーシックによるソウトウェア資産の継承を強調したBASIC/98系が，それぞれの特徴になる．

これから各メーカーが，標準規格に準拠した上で独自の機能を付加したベーシックの製品開発をすることになろう．そのときには，ニューベーシックが標準ベーシックと呼ばれるにちがいない．

参考資料

1)小平邦彦編(1991)数学II．東京書籍

2)市川新(1984)アメリカ国民の義務となったコンピュータ学の修得．コンピュータ＆ネットワーク4月号

3)Birnes, WJ ed.（1989）High-Level Languages and Software Applications．McGraw-Hill

4)市川新(1986)BASICで学ぶPROLOGシステム．啓学出版

5)草薙裕(1985)自然言語とコンピュータ言語．講談社

6)松田健生訳，市川新解説(1990)バック・トゥ・BASIC．啓学出版

7)ANSI（1987）Programming Languages − full BASIC．(ANSI X3.113-1987)

8)ANSI（1989）Modules and Individual Character Input for full BASIC．(ANSI X3.113a-1989)

9)西村恕彦他編(1990)アメリカ規格Full BASIC．共立出版

（注：本稿のプログラムリストは，実際の商品化された処理系向きに記述されたものではない．）

BASIC プログラミング用語集
アンダーフロー　計算結果が非常に小さな値になり，コンピュータの内部で処理できなくなること

特集……

やっぱり僕らはBASIC!

N88BASIC から QuickBASIC へ

構造化プログラミングを学ぶ

塚本吉彦

Beginner's
All-purpose
Symbolic
Instruction
Code

現在発売されているBASICの多くが，構造化言語仕様を備えている．パソコンの処理速度が遅く，メモリも少なかった頃にはプログラムも短くてすんだが，パソコンの速度が高速になり，メモリの量が増加するにつれてプログラムのサイズは大きくなり，その構造はより複雑になってきた．こうなると，構造化されていないプログラミング言語である旧BASICでは複雑なシステムのデバッグ作業など対処できなくなってしまったのである．これを解決する手段として発展してきたのが，構造化プログラミングだ．
構造化プログラミングの特徴は，プログラムの分岐構造を把握しやすく，プログラムが読みやすい点にある．ここでは，代表的な構造化BASICであるQuickBASICを取り上げ，構造化への移行のためのポイントと構造化とは何かを見てみる．

現在では，BASICも構造化BASICが主流となっている．BASICというとN88BASICを思い出すが，現在主流となりつつあるのが，QuickBASICやTrue BASICそれにVisual Basicだ．とくに，Visaul Basicにおいては，ほかの言語（CやPascal）では実現しえないような，高度なWindowsプログラミング環境を備えている．

これら，現在主流になりつつあるBASICは，基本線において共通した構造化言語仕様をもっている．ここでは，構造化BASICを題材に，構造化への移行のために何をしなければならないか，構造化とは何かを見てみる．

構造化プログラミングとは何か

プログラムをより見やすく，理解しやすく記述するのが構造化プログラミングだ．コンピュータの速度向上とともに，1つのアプリケーションの中で作成しなければならないコードの量は増加してきた．パソコンが遅かった時代には，ユーザーはパソコンに多くを望まなかった．だから，プログラムも短くてすんだ．パソコンの速度が高速になるにつれ，プログラムのサイズは大きくなり，それに応じて必要なメモリの量が増加し，アプリケーションが複雑になってきたのである．

こうなってくると，構造化されていないプログラミング言語である，BASIC，COBOL，FORTRANでは，複雑なシステムのデバッグ作業に対処できなくなってしまった．そこで，発展してきたのが構造化プログラミング言語だ．その代表は，C言語やPascalである．とくにPascalは，構造化プログラミングを教えるために作成された言語である．そして，BASICも，その影響を受け構造化プログラミング言語に変わってきたのである．

構造化プログラミングの特徴は，プログラムの分岐構造を把握しやすいという点にある．かんたんにいえば，プログラムが読みやすく理解しやすいのだ．読みやすいということは，とりもなおさずデバッグしやすいということである．また，理解しやすいということは，同じ理解力をもった人間が，より複雑なコードを作成できるということである．

認識論という学問に，マジックナンバ7±2というのがある．人間が同時に認識できる要素の数は人によって異なるが，だいたい5から頭のいい人で9であるというわけだ．鳥の中でも頭のよいカラスは3つまでわかるというから，人間の7つというのはたいしたものである．余談だが，もし，1人の人間が無制限に認識管理できるなら，大企業の社長さんは1人で社員全員の指揮管理ができるということになる．だが，実際には社長さんが管理できるのはせいぜい5人の担当取締役というわけだ．それ以上取締役のいる企業では，実際に力のある取締役が7人以下でほかが名前だけであるか，あるいは社長が取締役会を指揮できていないかのどちらかであるということになる．

この人間の認識の限界ゆえに，企業は構造をもち，プログラムも構造化されなければならないのだ．考えてみてほしい，もし，コンピュータがプログラムを作成するのであれば，なにも構造化などしなくてよいのである．実際問題，最適化コンパイラの出力するコードは，構造化など関係なく，とにかく速く実行できるコードを出力するのである．ではどのようにすれば読みやすくなるのかを考えてみよう．

プログラムを読みやすくするには？

まず，第1にモジュールわけである．N88BASICを無理に利用するのであれば，これはRUNコマンドや，GOSUBステートメントの利用になる．QuickBASICを利用するのであれば，プログラムを複数のサブプロシージャに分割する作業と，複数のプログラムを複数のファイルに分割する作業になる．分類の目的は，ひとつひとつのモジュールが管理できないほど大きくならないようにすることと，探しているモジュールがどこにあるのか，すぐにわかるようにすることである．この2つは大切である．1つ目のポイントの重要性はすでに述べてある．2つ目のポイントは，ものを探すには時間がかかるので，モジュールは，そのモジュールがあると思われるファイルに格納しなければならないということだ．たとえば，File.BasとGraph.Basという2つのファイルがあって，その中にDrawItemOnScreenというプロシージャがあったとしよう．このプロシージャがFile.Basに入っていると最悪である．普通，知らない人間は，Graph.Basを探す．このように，どこになにを入れるかも構造化の大切なポイントである．これができないなら，構造化をしないほうがよい．

第2は，構造化プログラミングの手法を利用することである．制御構造が明確になるようにする第1歩は，プログラムの中からGOTOを取り除くことである．約15年ほど前，プログラミ

BASICプログラミング用語集
入れ子　サブルーチンの内部でさらにサブルーチンを呼んだり，ループの中にループが存在すること．BASICではFOR〜NEXTの入れ子が使用される．

ング言語としてFORTRANやCOBOLが主流だったころ，GOTO文によりプログラム全体の構造が見えなくなるということがさかんに話題となった．その時登場したのが，この構造化プログラミングという技術である．

プログラムの中のあらゆる制御構造は，GOTOを用いなくても実現できる．であるから，プログラムの中にGOTOを記述する必要はない．GOTOを記述すると，その飛び先がどこにあるのか探さなければならない．GOTOの飛び先は，GOTOより前かもしれないし，あとかもしれない．ラベルが利用されている場合は，そのラベルを探すのがたいへんな作業となる．また，行番号でGOTOする場合，その分岐先の処理内容がまったく見当がつかないので，その先の処理を全部追いかけないと，全体のプログラムの様子がわからないのである．

実際の例として，リスト1のような処理を見てみよう．

この2つのコーディングは意味は同じだ．上は従来のN88BASICの記述方法，下はQuickBASICの記述方法である．N88BASIC方式では，A＝Bが成立する場合にどの部分が実行され，どの部分が実行されないのかを理解するには，注意深く行番号を追いかけなければならない．一方，下のQuickBASICの記述方法では，構造化され，プログラムが注意深くインデントされてコーディングされているので，A＝Bのとき，どこまでがスキップされるか一目瞭然である．同じことを理解するのに，必要な時間は上と下では何倍も違うのである．また，下のコーディングの場合，行番号は意味をもたない．したがって，行番号なしでプログラムを作成できるのである．行番号は，十分なエディタがなかったころの遺産である．このようなものを，現在も利用するのはたいへんバカげているとしか思えない．

変数名の付け方と定数の利用

筆者は，構造化を実現してよい結果を得るための重要な手法として，変数名の付け方と定数の利用があると考えている．第1に変数名の付け方である．最近はやりの付け方として，変数の型を変数名の最初にもってくる方法がある．BASICでは，変数の最後の文字で変数の種類を表現することもできるので，それでも十分だ．Windowsのプログラミングの中で使われているハンガリアン記法は，型を表すのにたいへん便利な方法の1つだ．変数の最初の文字を変数の型を表すのに用いるのである．

リスト2のように宣言すると，たいへん便利だ．また，大文字と小文字を区別して利用できる機能を利用して，Smalltalk風にTotalCostのように英語の単語を大文字ではじめて連結するのもはやっている．

Quic BASICで，定数を指定する場合には，Constというステートメントを使用する．たとえば，リスト3のようになる．

このように宣言すると，プログラム中でTableSizeという文字を45という数字のかわりに利用できる．たとえば，リスト4の2つのステートメントは同じ意味となる．

この定数による数字の指定は，大きなアプリケーションプログラムを作成する場合には，とても重要である．理由の第1は，プログラムをあとで読み返す場合に，読みやすくなるということである．リスト4の2つの例を見ても，上のほうがわかりやすいと思う．リスト5の例を見てほしい．

定数cWindowHightが，ウインドウの高さを表す定数だとして，その中心の座標をiCenterHightという変数に設定する例だ．最初のcWindowHight / 2は，ウインドウの高さの半分のところを設定したというのがよくわかる．しかし，残りの2つの例では，プログラムがなにをしているのか，プログラムのステートメントだけ見ても理解できない．定数の宣言は，めんどうだが大きなプログラムなど，あとで読

リスト1

```
1000 IF A=B THEN GOTO 1030
1010 ....
1020 ....
1030 ....

1000 If A <> B Then
1010     ....
1020     ....
1025 End If
1030 ....
```

リスト2

```
Dim iLoop As Integer
Dim sData As Single
```

リスト 3

```
Const cTableSize = 45
```

リスト 4

```
For I = 1 to cTableSize
For I = 1 to 45
```

リスト 5

```
Const cWindowHight = 120
Dim iCenterHight As Integer

iCenterHight = cWindowHight / 2

iCenterHight = 120 / 2

iCenterHight = 60
```

リスト 6

```
Const cWindowHight = 120
Const cWindowTop = 120

Center = cWindowHight / 2
Bottom = cWindowTop + cWindowHight
```

リスト 7

```
Const cWindowHight = 140
```

リスト 8

```
Center = 120 / 2
Bottom = 120 + 120
```

リスト 9

```
Center = 60
Bottom = 240
```

み返す可能性があるプログラムの場合，このように大きな力を発揮するのだ．

もう1つ，構造化プログラミングにとって定数の重要な用法は，プログラム中で同じ数値が複数利用されていて，場合によってはその数値が変わるかもしれない場合に使われる．

リスト6の例で，プログラミングの途中で都合によりWindowHightを120から140に変更したとする．もし，プログラムが，リスト6のようになっていたら，リスト7のように修正するだけですむ．しかし，リスト8のようになっていたら，修正は困難を極めるに違いない．

この場合は，検索機能を利用して120という数値を探し，その120がcWindowHightなのかcWindowTopなのかを判断しながら数値を変更すれば，まだ，なんとか処理できる．しかし，リスト9のようになっていたら，プログラムを全部読み直さなければならなくなる．

というわけで，定数について長く説明したが，このように，コーディング手法というのはプログラミングにおいてとても大切な技術なので，ぜひ利用するようにしてほしい．

構造体の利用

そのほかの重要な機能として，構造体の利用がある．QuickBASICでは，データをまとめて構造体として利用できる．構造体を利用すると，複数のデータをまとめて管理できるのでプログラムの複雑さを解消するのにとても優れた方法である．実際には，リスト10のようにする．

同じようなデータが多数ある場合，それぞれを別々に定義するとたいへんな操作になる．たとえば，リスト10の

BASIC プログラミング用語集
インタプリタ　プログラムをステップ単位で解析し実行する．ソースファイルの修正が，即ちプログラムの修正となるが，ステップ単位の解析のため実行速度が遅くなる．

定義をリスト11のようにばらばらにすると収拾がつかなくなる.

構造化とは プログラムを見える ようにする技術

このように，構造化とは，プログラムの構造を見えるようにする技術である．ここで見えるとは，プログラムの構造が，それをデバッグ，メンテナンスする自分の脳味噌で管理できるようにする技術である．このように考えると，結構重要なのは，プロシージャの作り方や変数の用意のしかたなどであることがわかる．

構造化プログラミングというと，Doや Select を使えばよいと思われるかもしれない．だが，そうではないのである．たとえば，If...Then...ElseIf...Then...ElseIf...EndIf を用いれば，大部分の Select ステートメントはわざわざ Select を用いなくても実現できる．これでは，意味がない．Select は Select Case にふさわしい用途の場合には，必ず使用しなければならない．たとえば，1つの変数の値によって分岐が複数行われる場合には，If..Then...ElseIf は用いてはいけないのである．

正しいプログラミングの方法は，とくにきまっているわけではないが，価値のあるプログラムを作成するためには，プログラムがある程度のコーディング量になるのはしかたがないことであると思う．ある程度のコーディングを行うために，このように構造的なプログラムを記述するようにして欲しい．

リスト 10

```
Type Hiduke
        Nen As Integer
        Tsuki As Integer
        Hi As Integer
End Type

Dim Kino As Hiduke
Dim Kyo As Hiduke
Dim Ashita As Hiduke
```

リスト 11

```
Dim KinoNen As Integer
Dim KinoTsuki As Integer
Dim KinoHi As Integer
Dim KyoNen As Integer
...
```

COLUMN

QuickBASIC の解説書

QuickBASIC4.5プログラミング入門
志村拓著／アスキー／2,980円
免許皆伝／QuickBASIC プログラミング道場
木村忠文著／アスキー／1,800円
QuickBASICの便利な活用本
手越義昭・新長輝則著／井上書院／2,400円
QuickBASIC 初級プログラミング入門（上・下）
河西朝雄著／技術評論社／各2,100円
QuickBASIC 中級プログラミング
小池慎一・森博著／技術評論社／2,500円
QuickBASIC入門
吉川敏則・播磨央著／啓学出版／2,500円
QuickBASICテクニカルマニュアル
伊東ひろみ著／新星出版社／1,600円
QuickBASIC4.5ステートメント＆関数リファレンス
マイクロソフト（株）監修／翔泳社／2,400円
QuickBASICVer.4.5プログラミングブック
塚本吉彦著／翔泳社／2,000円
QuickBASICスタイルブック
イント・ツー・ワン著／翔泳社／2,300円
QuickBASICプログラミング
野口雄平・金田知之・高木聡著／ソフトバンク／2,000円
QuickBASICプログラムブック
塚越一雄／電波新聞社／1,800円
QuickBASICハンドブック Vol.1［基本操作］
阿部友計著／ナツメ社／1,000円
QuickBASICハンドブック Vol.2［リファレンス］
阿部友計著／ナツメ社／1,300円
QuickBASIC入門
イント・ツー・ワン著／日本実業出版社／1,600円

特集……

やっぱり僕らはBASIC!

プロジェクト指向環境を備えた
先進のBASIC

Microsoft BASIC Ver.7.1

海老原浩之

Beginner's
All-purpose
Symbolic
Instruction
Code

この頃やたらとBASIC言語に触れる機会が多い．Windowsではvisual Basic，DOS&OS/2環境ではMS-BASIC Ver.7.1やTrue BASICなどである．またプログラムの開発においては，まだBASICのほうが使いやすいと聞くこともある．ここでは，今注目を集めているBASICの1つとして，本誌7月号で新製品紹介として取り上げたMS-BASIC Ver.7.1をとくに構造化プログラミングとモジュール化，新規に採用された機能などに重点をおいて取り上げてみる．

MS-BASIC Ver.7.1の概要

まず最初に簡単にMS-BASIC Ver.7.1の概要をおさらいしておく.

- プロジェクト指向
- EMSをサポートした大規模プログラムの作成
- ビジネスプログラミング支援
- MS OS/2のサポート
- 強力なオプティマイズ機能

上記の5つはMS-BASIC Ver.7.1になってからの特徴だが,実際にはQuickBASICでおなじみの特徴もMS-BASIC Ver.7.1には含まれている.それは,

- モジュール化の概念
- 構造化プログラミング
- 複数言語によるプログラミング

などの特徴である.では,この8つの特徴を簡単に説明していく.

プロジェクト指向

QuickBASICから採用された統合環境によるプログラム開発のことをプロジェクト指向という.MS-BASIC Ver.7.1では,従来のQuickBASIC環境を拡張したQuickBASIC Extended VersionとMS-C Ver.6.0で採用されたProgrammer's Work Bench(PWB)からの操作を中心としてCodeView,NMAKE,ソースブラウザなどが使える環境の2つが提供されている.

EMSのサポート

EMSを利用できるようになったことで,従来までの640KBのメモリ制限がなくなった.それによって,最大16MBまでのアプリケーションを実行可能になり,PWBやQBX環境化においてもプログラムの開発やデバッグが行いやすくなった.

ビジネスプログラミングの支援

ISAMファイル,ツールボックスライブラリ(ユーザーインターフェース,プレゼンテーショングラフィックス,フォント,行列演算)がサポートされた.これらのISAMエンジンやツールボックスがライブラリとしてサポートされたことによって,ビジネスアプリケーションが作成しやすくなった.

MS OS/2のサポート

いままでのBASIC言語では,OS/2のプログラムを作成することができなかったが,このMS-BASIC Ver.7.1によって可能になった.もちろん,OS/2のAPIを使ったプログラミングも可能である.

強力なオプティマイズ

MS-C Compilerと同じオプティマイズ技法を取り入れたことによって,従来のQuickBASICよりもコンパクトで高速な実行モジュールを生成できるようになった.

モジュール化の概念

従来のサブルーチンやユーザー定義関数を拡張することによって,プロシージャの概念を取り入れた.それによってモジュール化プログラミングが可能になり,開発効率をあげることができるようになった.モジュール化されたサブルーチンやユーザー定義関数には,LOCAL変数やSTATIC変数もサポートされ再起呼び出しも可能になっており,引数の受け渡しやファンクションでは値を返すこともできる.1度作成されたプロシージャは他のプログラムから利用することも可能である.

さらに,作成されたサブプログラムやファンクションはBASIC内の関数やステートメントと同じように使うことが可能である.

構造化プログラミング

構造化のための制御文,宣言文を取り入れたことによって,GOTO文を多用することなどなく,ブロックごとに構造化された見やすいプログラムを書ける.

複数言語によるプログラミング

MS-C,MASM,MS-FORTRANなどのマイクロソフト社製の他の言語で作成されたプロシージャをMS-BASICから呼び出すことができる.もちろん引数や戻り値の受け渡しも可能である.これによって,他の言語で作成された資産をMS-BASIC Ver.7.1で利用することが可能である.

構造化BASIC

QuickBASICでもそうだったが,MS-BASIC Ver.7.1が従来よりも

BASICプログラミング用語集
エミュレータ　たとえば数値演算コプロセッサをもたないコンピュータ上で浮動小数点演算を行うためなど,ソフトウェアでハードウェアの機能を実現すること.

使いやすくなったのは，ブロック化ステートメント，プロシージャ，モジュール化といった概念を取り入れたことにより，構造化BASIC言語に変わったためである．つまり，従来までのMS-BASIC Ver.5.3やN88BASICに代表されるような行番号が付いた見づらいプログラムを書かなくてもすむようになったのだ．これによって悪評高いスパゲッティプログラムにならなずにすみ，機能別にモジュール化したプログラムを書けるためデバッグも容易になり，プログラム自身も非常に整然とした形になった．また，作成されたモジュールを別プログラムから利用できるなど，大規模なアプリケーションの作成をすることも可能になったといえる．

さらに1度作られたモジュールは，ユーザー独自のライブラリとして作成することも可能であるため，C言語のように独自のライブラリ資産を増やしていくことも可能である．

では，実際に構造化BASICのプログラムを交えながら話を進めていくことにしよう．

構造化ステートメント

構造化ステートメントとして以下のものがある．

- DO....LOOP
- FOR...NEXT
- WHILE...WEND
- SELECT CASE...END SELECT
- IF ［ブロック］ END IF
- TYPE...END TYPE

これらの中には従来からあるステートメントもあるし，新規に追加されたものもある．たとえば新規に追加されたものに，SELECT CASE...END SELECT，TYPE...END TYPEがある．このSELECT CASE ...END SELECTはC言語を使ったことがある人は馴染みがあるだろう．このステートメントを使うことで，従来のIF..THEN文を多用した判定処理を1つのブロックとしてまとめることができるのでIF..THEN文の多用をさけることができる．TYPE...END TYPEによって，FILED文で定義していたレコードの定義を構造体として定義できるようになる．これによって横に長々と書いていたFILED文によるレコード定義も，TYPE...END TYPE文によって縦に並んだ

リスト1　DO ... LOOP & SLECT CASE ... END SELECTのサンプル

```
100 INPUT A
110 IF A=1 THEN PRINT "A"
120 IF A=2 THEN PRINT "B"
130 IF A=3 THEN PRINT "C"
140 IF A=9 THEN GOTO  160
150 GOTO 100
160 END
      |
      |
      v

DO
  INPUT A
  SELECT CASE A
       CASE 1
            PRINT "A"
       CASE 2
            PRINT "B"
       CASE 3
            PRINT "C"
       CASE 9
            EXIT DO
  SELECT END
LOOP
END
```

リスト2　IF ［ブロック］ END IFのサンプル

```
100 A%=6350
110 INPUT A$
120 IF A$="YEN" THEN PRINT "¥";:PRINT USING "###,###";A% ELSE PRINT "$";:PRINT USING "###,###"
;A%

    |
    |
    v

A% = 6350
INPUT A$
IF A$ = "YEN" THEN
      PRINT "¥";:PRINT USING "###,###";A%
   ELSE
      PRINT "$";:PRINT USING "###,###";A%
END IF
```

BASICプログラミング用語集
OS　コンピュータのハードウェア・ソフトウェアを有効に利用するための総合的管理を行うソフトウェア．さらにタイムシェアリングなどマルチタスク管理を行うOSもある．

リスト 3　TYPE . . END TYPE のサンプル

```
100 FIELD #1,6 AS A$,16 AS B$,16 AS C$
      |
      |
      v

TYPE FItem
          A AS STRING * 6
          B AS STRING * 16
          C AS STRING * 16
END TYPE
```

見やすいものになり，コメントも付けやすくなったのでレコード内の項目もわかりやすくできる．FOR...NEXT，WHILE...WENDなどは従来のBASICでもあったステートメントだが，今回多少改良されたものにIF［ブロック］END IFがある．従来のIF...THEN文をブロック化して書けるようにしたもので，従来のIF文のように横長にならずにすむようになった．

このように言語仕様自体が見直され，構造化されたことによって制御構文を書きやすく見やすいものにしている．制御構文に関していえばC言語に近づいたといえる．

モジュール化

従来のBASICでは，サブルーチンやユーザー関数を定義するとしたらGOSUB文やDEF FN文を利用するしかなかった．しかも，同じような処理をするサブルーチンやユーザー関数があっても複数のプログラムからの利用はできなかった．いわば1つのプログラム内でしか利用できない閉じた世界の状態であった．

しかし，QuickBASICやMS-BASIC Ver.7.1から，プロシージャの概念が取り入れられたことによって，閉じた世界が開かれた世界に変わったのである．同じような処理をするサブルーチンやユーザー定義関数は，モジュール化することで複数のプログラムから利用することができるようになった．つまり，モジュール化によってサブルーチンやユーザー定義関数の独立性が増し，従来のような長いスパゲッティのようなプログラムにならなくなったのである．さらにモジュール化されたサブルーチンやユーザー定義関数は，部品単位として扱うことができるのでデバッグなども容易になり，プログラムの保守性もよくなっている．当然作成されたモジュールはライブラリ化してしまえば，どのプログラムからも利用できるのでプログラムの生産性向上につながる．

モジュールの使用法として，FUNCTION...END FUNCTION，SUB...END SUBのサンプルを示しておく（リスト4）．

注：モジュールとは，個別にコンパイルが可能なBASICのソースを意味する．それをコンパイルしてできあがっ

リスト 4　FUNCTION ... END FUNCTION & SUB ... END SUB のサンプル

```
DECLARE SUB DISPLAY (MOJI.DATA$, YP!, XP!)
        .
        .
A$="TEST":Y=10:X=10
CALL DISPLAY(A$,Y,X)
        .
        .
SUB DISPLAY (MOJI.DATA$, YP, XP)

        LOCATE YP, XP: PRINT MOJI.DATA$

END SUB


DECLARE FUNCTION ACCEPT$(L%,X%,Y%)
        .
        .
X$ = ACCEPT$(5,10,7)
PRINT X$
        .
        .
FUNCTION ACCEPT$ (L%,X%,Y%) STATIC

        LOCATE Y%,X%
        ACCEPT$=INPUT$(L%)

END FUNCTION
```

表1　MISAMの仕様

データベースの大きさ	128MB
同時にオープンできるデータベース	4
1データベース中のテーブル数	無制限
同時にオープンできるテーブル数	13
1テーブルあたりに設定できるインデックス	29
キーフィールドの長さ	256バイト
複合インデックス数	10

表2　ISAMの操作命令

ファイル操作処理	
OPEN	データベースファイルのオープン
CLOSE	データベースファイルのクローズ
DELETETABLE	テーブルの削除
テーブル情報の取得	
BOF	先頭レコードの判定
EOF	最終レコードの判定
LOF	レコード数
インデックスの処理	
CREATEINDEX	インデックスの作成
GETINDEX$	現在のインデックス名を得る
SETINDEX	インデックスの指定
DELETEINDEX	インデックスの削除
レコードの処理	
DELETE	カレントレコードの削除
INSERT	カレントレコードの挿入
RETRIEVE	カレントレコードの読み込み
UPDATE	カレントレコードの更新
カレントレコードの相対的移動処理	
MOVEFIRST	インデックスの順で並んでいるレコードの先頭に移動
MOVELAST	インデックスの順で並んでいるレコードの最後に移動
MOVENEXT	次のレコードに移動
MOVEPREVIOUS	前のレコードに移動
トランザクション処理	
BEGINTRANS	トランザクション処理の開始
CHECKPOINT	現在のバッファ状態を強制的にディスクに保存する
COMMITTRANS	トランザクション処理の終了
ROOLBACK	SAVEPOINTで指定された以降までのトランザクション処理の取り消し
ROOLBACK ALL	すべてのトランザクション処理の取り消し
SAVEPOINT	トランザクション処理で，取り消し開始点の状態を設定
検索処理	
SEEKEQ	指定したキーの値と一致するレコードを検索
SEEKGE	指定したキーの値以上のレコードを検索
SEEKGT	指定したキーの値より大きいレコードを検索
その他	
TEXTCOMP	文字列の比較

たものがオブジェクトモジュールと呼ばれる．

ISAMファイルとツールボックス

MS-BASIC Ver.7.1では，従来までのBASICにはなかった新しい機能が加わった．それがISAMファイルのサポート，ユーザーインターフェースツールボックスのサポートである．これらが新しく加わったことによって，ビジネスユースに力をおいたBASIC言語になったといえる．

MS-BASIC Ver.7.1のISAMエンジンは，単純にISAMファイルを実現したというよりも，簡易データベースを構築できるといったものである．このISAMエンジンは，DOSの場合TSR(常駐型)とコンパイル時にリンクするライブラリ形式の2種類で供給され，OS/2ではDLL(ダイナミックリンクライブラリ)によって供給される．データは最大128MBまで扱うことができ，ISAMのインデックスオーダーはMicrosoft Excelの50音ソーティングオーダーに準拠している．このISAMの仕様追加によって，ISAMファイルを扱う命令が追加された．ISAMの仕様とISAMの命令に関しては表1，2に示しておくので参照してほしい．

次にツールボックスライブラリは，「ユーザーインターフェース，プレゼンテーショングラフィックス，フォント，行列演算」の4つの構成からなっている．このうちよく使われるのは，「ユーザーインターフェースとプレゼンテーショングラフィックス，フォント」の3つのツールボックスだろう．

BASICプログラミング用語集
オーバーレイ　大きなプログラムなどで，プログラムをモジュール単位で分割し，必要な部分のみをメインメモリ上にロードして使用する方法．

残った行列演算(二次元配列の演算)ツールボックスは，通常ではあまり使うことがないと思われる．

これらのツールボックスライブラリについて簡単に説明したい．

ユーザーインターフェース ツールボックス

ユーザーインターフェースツールボックスは，Microsoft WORKSやQuickBASIC/QuickCなどに使われているようなキャラクタ型のウィンドウインターフェースを簡単に構築できるライブラリである．たとえば，ドロップダウンメニューやダイアログボックスの表示，マルチウィンドウ表示などが簡単にプログラミングできるようになっている．

プレゼンテーショングラフィックスツールボックス（グラフ表現）

プレゼンテーショングラフィックスツールボックスは，解析されたデータを基にして円グラフ／縦棒グラフ／横棒グラフ／線グラフ／散布図などを簡単に描けるようにするライブラリである．

フォント ツールボックス

MS-Windowsのビットマップフォントを利用した文字を，グラフィック画面に表示させるライブラリである．ただしフォントは英数字に限られ，日本語フォントのサポートはされていない．

行列演算 ツールボックス

二次元配列(行列)の加減乗除やガウスの消去法を使った連立方程式の解法や行列式などの計算ルーチンを提供するライブラリである．

さて，ISAMやツールボックスライ

リスト5　ISAMの部分サンプルプログラム

```
DEFINT A-Z

'データテーブルの定義
TYPE FILETBL
     NO          AS INTEGER         ' 番号
     KANA        AS STRING * 16     ' カタカナでの読み
     NAME        AS STRING * 16     ' 漢字での名前
     ZIP         AS STRING * 6      ' 郵便番号
     ADDRESS1    AS STRING * 20     ' 住所 1
     ADDRESS2    AS STRING * 20     ' 住所 2
     ADDRESS3    AS STRING * 20     ' 住所 3
     TEL         AS STRING * 13     ' 電話番号
END TYPE

DIM FILEREC AS FILETBL

               .
               .
               .
               .
               .
               .

'データベースファイルがない場合は新規作成する
    OPEN "ADDRESS.MDB" FOR ISAM FILETBL "Customer" AS #1
    CREATEINDEX #1, "NOIdx", 1, "NO"
    CREATEINDEX #1, "KANAIdx", 0, "KANA"
    SETINDEX #1, "NOIdx"

               .
               .

'データベースが存在する場合のオープン処理
    OPEN "ADDFILE.MDB" FOR ISAM FILETBL "Customer" AS #1
    SETINDEX #1, "NOIdx"


'処理作業の選択
DO
    LOCATE 1, 5
    PRINT "A)追加 D)削除 U)修正 S)検索 C)検索項目の変更 E)終了 :  "
    LOCATE 1, 72: SEL$ = INPUT$(1): PRINT SEL$
    LOCATE 23, 10: PRINT "

    SELECT CASE UCASE$(SEL$)
          CASE "A"
               LOCATE 22, 10: PRINT "確認 [ ] Y/N"
               DO
                    CALL InputRec(FILEREC)
                    LOCATE 22, 16: KAKU$ = INPUT$(1): PRINT KAKU$
               LOOP WHILE UCASE$(KAKU$) <> "Y"
               INSERT #1, FILEREC
               CALL SeekRec(FILEREC)
          CASE "D"
               IF NOT EOF(1) AND NOT BOF(1) THEN
                  DELETE #1
               ELSE
                  BEEP
                  LOCATE 23, 10: PRINT "ﾃﾞｰﾀを削除出来ません"
               END IF

         CASE "U"
               IF NOT EOF(1) AND NOT BOF(1) THEN
                  LOCATE 22, 10: PRINT "確認 [ ] Y/N"
                  DO
                       CALL InputRec(FILEREC)
                       LOCATE 22, 16: KAKU$ = INPUT$(1): PRINT KAKU$
                  LOOP WHILE UCASE$(KAKU$) <> "Y"
                  UPDATE #1, FILEREC
                  CALL SeekRec(FILEREC)
               ELSE
                  BEEP
                  LOCATE 23, 10: PRINT "ﾃﾞｰﾀを更新できません"
               END IF

         CASE "S"
               SELECT CASE GETINDEX$(1)
                     CASE "NOIdx"
                          LOCATE 3, 10: INPUT "NO : "; FILEREC.NO
                          SEEKEQ #1, FILEREC.NO
                     CASE "KANAIdx"
                          LOCATE 3, 10: INPUT "読み : "; FILEREC.KANA
                          SEEKEQ #1, FILEREC.KANA
               END SELECT
               IF NOT EOF(1) THEN
                  CALL DispRec
```


オペランド　演算数ともいい，演算対象となる数値のこと．変数や定数，関数がBASICではオペランドとして使用できる．

ブラリを使ったプログラムだが，今までのBASICにない機能なのでプログラミングになれるまで多少なりとも時間がかかる．しかし，一度なれてしまえばデータの管理に関しては，ISAMの使用を前提とした設計が中心になり，ユーザーインターフェースやデータのグラフ化などは非常に楽に行える．

では，実際にISAM，ユーザーインターフェースツールボックスを利用した部分サンプルプログラムを紹介する．

EMS(ディスク)を利用したオーバーレイプログラム

MS-BASIC Ver.7.1では，実行プログラムの一部をEMSまたはディスクにオーバーレイさせることができる．オーバレイさせるプログラムは，LINK時に指定するだけでよい．

LINK A ＋ B ＋ (C)

とすれば問題なく行える．この場合，AとBの2つのモジュールは通常のメモリに常駐し，CモジュールがEMSまたはディスクにオーバーレイするようになるわけである．またオーバーレイには制約があるので，その制約を守った状態でオーバーレイするモジュールを作らなければならない．

オーバーレイの制約一覧

- 各モジュールの最大サイズは256KBまでで，各プログラムのオーバーレイの数は最大64まで．
- オーバーレイモジュールはLINKのコマンドラインで最初に指定できない．
- プログラムのオーバーレイバージョ

```
            ELSE
               BEEP
               LOCATE 23, 10: PRINT "ﾃﾞｰﾀが見つかりません"
            END IF

       CASE "C"
            LOCATE 3, 10
            PRINT "検索項目の変更 1) ID番号  2) 読み :";
            n$ = INPUT$(1): PRINT n$
            SELECT CASE n$
                   CASE "1"
                      IF GETINDEX$(1) <> "NOIdx" THEN SETINDEX #1, "NOIdx"
                   CASE "2"
                      IF GETINDEX$(1) <> "KANAIdx" THEN SETINDEX #1, "KANAIdx"
            END SELECT
         ' 先頭レコードの表示
        IF NOT EOF(1) AND NOT BOF(1) THEN CALL DispRec

      END SELECT
LOOP WHILE UCASE$(SEL$) <> "E"

CLOSE #1
END

                .
                .
                .
                .

'読み込まれたテーブルデータの内容を表示する
SUB DispRec
    DIM REC AS FILETBL
    RETRIEVE #1, REC
    LOCATE 5, 19: PRINT USING "####"; REC.NO
    LOCATE 7, 19: PRINT REC.KANA
    LOCATE 9, 19: PRINT REC.NAME
    LOCATE 11, 19: PRINT REC.ZIP
    LOCATE 13, 19: PRINT REC.ADDRESS1
    LOCATE 15, 19: PRINT REC.ADDRESS2
    LOCATE 17, 19: PRINT REC.ADDRESS3
    LOCATE 19, 19: PRINT REC.TEL
END SUB

                .
                .
                .
                .

' カレントインデックスでデータを検索する.
SUB SeekRec (requred AS FILETBL)
        SELECT CASE GETINDEX$(1)
          CASE "NOIdx"
               ' ID フィールドは重複のないインデックスのため,
               ' "等しい"条件で検索が行える.
               SEEKEQ #1, requred.NO
          CASE "KANAIdx"
               ' カナ読みフィールドのインデックスは重複が許されるため, 最初に
               ' 一致するレコードを見つけた後, 順次 ID を比べ, 目的のレコードを
               ' 検索する.
               SEEKEQ #1, requred.KANA
               RETRIEVE #1, REC
          DO WHILE requred.NO <> REC.NO
             MOVENEXT #1
             RETRIEVE #1, REC
          LOOP
        END SELECT
END SUB
```

リスト６　ユーザーインターフェースプログラム部分サンプル

```
'ユーザーインターフェースツールボックスを使ったプログラムを作成するには
'以下の項目は、先頭で必ず宣言しなければならない

'$INCLUDE: 'qbx.bi'
'$INCLUDE: 'general.bi'
'$INCLUDE: 'mouse.bi'
'$INCLUDE: 'menu.bi'
'$INCLUDE: 'window.bi'

COMMON SHARED /uitools/ GloConst           AS HardConstType
COMMON SHARED /uitools/ GloMenu            AS MenuMiscType
COMMON SHARED /uitools/ GloTitle()         AS MenuTitleType
COMMON SHARED /uitools/ GloItem()          AS MenuItemType
COMMON SHARED /uitools/ GloWindow()        AS windowType
```

BASIC プログラミング用語集
拡張子　ファイル名の中でピリオドからうしろにある部分．MS-DOSの場合，拡張子によってファイルの種類が判断できる．

```
COMMON SHARED /uitools/ GloButton()       AS buttonType
COMMON SHARED /uitools/ GloEdit()         AS EditFieldType
COMMON SHARED /uitools/ GloStorage        AS WindowStorageType
COMMON SHARED /uitools/ GloWindowStack()  AS INTEGER
COMMON SHARED /uitools/ GloBuffer$()

DIM GloTitle(MAXMENU)            AS MenuTitleType
DIM GloItem(MAXMENU, MAXITEM)    AS MenuItemType
DIM GloWindow(MAXWINDOW)         AS windowType
DIM GloButton(MAXBUTTON)         AS buttonType
DIM GloEdit(MAXEDITFIELD)        AS EditFieldType
DIM GloWindowStack(MAXWINDOW)    AS INTEGER
DIM GloBuffer$(MAXWINDOW + 1, 2)

              .
              .
              .

' ================================
' メニューの設定と初期化.
' ================================

'メニュー表示の初期化
       SCREEN 0
       MenuInit
       WindowInit
'メニューの定義
       MenuSet 1, 0, 1, "File", 1
       MenuSet 1, 1, 1, "Open   F1", 1
       MenuSet 1, 2, 1, "Close  F2", 1
       MenuSet 1, 3, 1, "-", 0
       MenuSet 1, 4, 1, "EXit   Ctrl+X", 1
       MenuSet 2, 0, 1, "Help", 1
       MenuSet 2, 1, 1, "About  F10", 1
'メニューで使うショートカットキーの定義
       ShortCutKeySet 1, 1, CHR$(0) + CHR$(59)
       ShortCutKeySet 1, 2, CHR$(0) + CHR$(60)
       ShortCutKeySet 1, 4, CHR$(24)
       ShortCutKeySet 2, 1, CHR$(0) + CHR$(68)
'メニューの色設定
       MenuColor 0, 3, 14, 0, 3, 0, 14
'メニューの前処理
       MenuPreProcess
'メニューの表示
       MenuShow
'マウスの表示
       MouseShow

              .
              .
              .

' =========
' メイン
' =========

        Finished = FALSE
        WHILE NOT Finished
                kbd$ = MenuInkey$      'キー入力のチェック
                menu = MenuCheck(0)    'メニュー番号チェック
                item = MenuCheck(1)    '項目番号チェック
                SELECT CASE menu
                  CASE 1
                       SELECT CASE item
                              CASE 1
                              FileOpen
                              CASE 2
                              FileClose
                              CASE 4
                              Finished = TRUE
                       END SELECT
                  CASE 2
                       SELECT CASE item
                              CASE 1
                              HelpAbout
                       END SELECT
                  CASE ELSE
                END SELECT
        WEND

' ====================================================================
' プログラムの終了.
' ====================================================================

        MouseHide
        COLOR 7, 0
        CLS
        END

              .
              .
              .
```

ンを作成するとき，プログラム中の各モジュールを互換性のあるオプションを指定してコンパイルする必要がある.

・オーバーレイを使うプログラムでリンクをするとき，/PACKCODEは使えない.

・オーバーレイすることができるモジュールは，8086long（32ビット）呼び出し・復帰によって制御が引き渡されたり戻されたりするものに限る．また，LINKは関数ポインタを使って間接的に呼び出されるオーバーレイモジュールを作成できない．ポインタを使って関数の呼び出しを行う場合は，同じオーバーレイかルートにおく必要がある.

OS/2における MS-BASICの活用

MS-BASIC Ver.7.1において，OS/2に対応したアプリケーションの開発も可能となった．MS-DOSのメモリ制限などは気にせずに大規模なアプリケーション開発が行え，C言語以外にもOS/2用の開発言語がでたことはユーザーにとって非常にありがたいことである．世の中にはまだまだ，CよりもBASICのほうが使いやすいといったユーザーも多いだろう．これは，いままでC言語は苦手だがBASIC言語ならできるといったユーザーも含めて，OS/2アプリケーション作成への道を開いてくれるものである.

また，簡単なツールやプログラムの作成ならば，C言語よりもBASICのほうが適している．簡単に何かを作成するといったことが，OS/2で実現できることはOS/2上で作業をする場合においても都合のよいことではなかろうか．さらに，既存のBASICで作成されているプログラムに関しても，OS/

BASIC プログラミング用語集
型　変数の構造のこと．BASICの変数には数値変数や文字変数があり，さらにそれぞれが細かく定義されている．またユーザー自身が定義することもできる．

2に対応した修正をすれば動作させることができるのである．BASICからC言語に移植する手間と時間を考えれば，遥かに効率がよいといえる．

OS/2対応のアプリケーション開発が容易

開発効率に関しても，PMに対応したアプリケーションの作成を望まないのであれば，BASICのほうが遙かによいだろう．C言語では，OS/2のAPIを使用したプログラミング作成が基本となる．そうなれば，既存のC言語の学習に加えてOS/2のAPIの勉強をしてからでないとOS/2のアプリケーションの作成はできない．しかし，MS-BASIC Ver.7.1においてはOS/2のAPIを多少勉強しなければならない部分もあるが，基本的にはBASICが本来もっている命令だけでOS/2に対応したアプリケーションを作成することができる．

MS-DOSのプログラムが活かせる

基本的には，MS-DOS上で開発したMS-BASIC Ver.7.1のアプリケーションがそのまま動作する．しかし，マシン語を利用してハードウェアを直接操作するようなプログラムや，サウンド関数を使用したプログラムに関しては，そのままではOS/2上では動作しないので注意する必要がある（表3）．

MS-BASIC Ver.7.1においては，通常のBASICの命令に加えてOS/2のAPIを使用したプログラムの作成も可能である．OS/2におけるすべてのAPIが使えるわけではないが，DOSxxxやMouxxxといったプロセスコントロール，セマフォ，シグナル，パイプやキュー，エラーハンドリング，セッションマネージャ，ファイル管理，マウスなどのファンクションを利用することができるようになっている．これらのOS/2 APIを利用するには，OS/2 APIを利用可能にするインクルードファイルが必要である．インクルードファイルについては表4にまとめてみたので参照してほしい．また，このOS/2のAPIの使用方法も通常のBASICの命令とさして変わるものではないので，APIがどのような働きをするのかさえ理解すればまったく問題なく使いこなさせるだろう．

表3 DOSから移行する場合OS/2で使えない文と関数

Int86
Int86X
Int86Old
Inyerrrput
INP
IOCTL
IOCTL$
OUT
PLAY
SETMEM
SOUND
WAIT

すべてのグラフィックス関係の文と関数

表4 OS/2 APIファンクションを使用するためのインクルードファイル

BASEDOSPC.BI	プロセスコントロール，セマフォ，シグナル，パイプおよびキュー エラーハンドリングとメッセージ，セッションマネージャ
BASEOSFL.BI	デバイスドライバ，ファイル管理
BSESUBMO.BI	マウス操作
BASEOSPE.BI	各国別設定，リソース管理，モジュール管理，日付およびタイマメモリ管理，情報セグメント

リスト7 OS/2 APIファンクションCALLのサンプル

```
'ビープ音をならします.

'$INCLUDE:'BASEDOSPC.BI'

FOR I=1 TO 10
'                 v・・・・・継続時間(ミリ秒)
   X=DosBeep(600,175)
'               ^..........周波数(ヘルツ)
   X=DosBeep(1200,200)

NEXT I
（注意）＊OS/2のAPIファンクションを使う場合，そのファンクションの使い方を知るためにOS/2のプロ
グラマーズリファレンスマニュアルが必要になります.
```


BASICプログラミング用語集
型チェック　演算子に対してオペランドが正しいか，または関数に渡す引数が関数で定義されている型と一致するかを調べること．

特集……

やっぱり僕らはBASIC!

512倍簡単になるWindowsプログラミング

Microsoft VisualBasic

酒井法雄

Beginner's
All-purpose
Symbolic
Instruction
Code

VisualBasicは，Windows上で動作するプログラムを開発するためのプログラミング言語＋プログラミング環境である．したがって，いちがいにその文法的な特徴をほかのBASIC処理系と比較することはできない．しかし，VisualBasicを使ったWindowsプログラミングは，従来のSDK＋Microsoft Cよりははるかに簡便であるとの評判は高く，現状ではまだ日本語対応バージョンすら発売されていないにもかかわらず，Windowsプログラミングをいっそう身近なものにしてくれる処理系として注目を集めている．本誌11月号特集「Windowsプログラミングの作り方」では，VisualBasicの開発環境を中心に取り上げたが，本稿では複数のサンプルプログラムをあげて，その作成手順を示し，VisualBasicによるプログラミングの実際を紹介する．

QuickBASICのヒットで，「まだまだBASICもいけるぞ」と思ったのも束の間，巷ではMS-Windows 3.0が大騒ぎです．「またしても，BASICはおいてきぼりか……」と思っていたら，出てきたのがVisualBasic．これで，WindowsのプログラミングもBASICでOKです．

Windows 3.0のアプリを作りたい

MS-Windowsはずいぶん前からある商品なのですが，バージョン3.0になってはじめて脚光を浴びるようになりました．というのも，Windowsの環境であるGUI（グラフィカルユーザーインターフェース）には，多くのメモリとCPUパワーを必要としたからです．時代がハードウェアを進歩させ，Windowsはそれに合わせるように機能を拡大し，やっと満足のいく環境を作ることができたのです．アメリカでは数カ月で300万本が売れたということで，わが国でも鳴り物入りで登場したWindows 3.0ですが，日本語に対応したWindows 3.0アプリケーションソフトも少なく，「買ったはいいが動かすソフトがない」というのが現状でした．徐々にWindows 3.0対応ソフトも出てきてはいますが，まだまだ少ないようです．

これには，いくつか原因があります．以前までのWindowsは，あくまでマイクロソフト社自体のプレゼンテーションソフトという戦略商品でしかなく，実用に供する製品ではないとわが国では認識されてきました．したがって，Windowsのプログラミングができる人自体が少なかったのです．需要がなければ供給もないわけで，Windowsプログラミングの参考となるような文献もあまり見あたりませんでした．

アメリカでのWindows 3.0のヒットを見て，「次はWindowsだ」と思った人も少なくはなかったはずなのですが，ここにも落し穴がありました．Windows対応のアプリケーションを作るには，MicrosoftCとSDK（ソフトウェアデベロップメントキット）が必要なのです．SDKとは，Windowsのプログラムを作るための膨大な数の専用関数や，開発に必要なツールを集めたものです．この関数やツールも，Windows 3.0に合わせて多数の変更や追加が行われています．アメリカで売っているSDKを買っても，日本語仕様に合わせた変更はしてありませんし，ドキュメントも全部英語です．これでは，Windows 3.0の発売と同時に多数のアプリケーションが発売されるというわけにはいきません．

さらに大きな問題は，SDKもMicorsoftCも高価であることです．両方を買うと，定価ベースで20万円ほどの出費になってしまいます．これでは，アマチュアのプログラマが楽しんでプログラミングし，PDSやフリーソフトウェアとして流通させるなどということはなかなかできません．

また，Windowsのプログラミングには，独自の概念と多数の関数やメッセージなどの知識が必要です．また，Cのソースファイルだけあればいいというものではなく，ウィンドウやビットマップ，カーソル，アイコンなどのリソースや，それを記述したファイルなど，多数のファイルを扱わなくてはなりません．しかも，開発中はMS-DOSとWindowsの間を何度も行き来しなくてはなりません．ウィンドウを出してその中に「Hello World」という文字列を表示するだけのプログラムでも，100行ほどのソースファイルになります．これは，Windowsのためのおまじないのようなコードをたくさん書かなくてはならないからです．

こんな状態では，お金が余っていても「Windowsのプログラミングなんかしたくないっ!」と思うのが人情でしょう．しかし，一方で「Windowsのプログラミングができたら楽しいだろうなぁ」とも思ってしまいます．なんたって，流行の最先端ですから．

VisualBasic登場

しかし，よく考えると妙な話です．MS-DOSではいろいろなおまじないみたいなコマンドがいっぱいあって，それをおぼえないとちゃんと使いこなせませんでした．MS-DOSを使いこなしたいという需要にあわせ，MS-DOSの入門書が世の中には氾濫したものです．なんとかMS-DOSが使えるようになっても，その上で動いているアプリケーションはみんなバラバラな操作性でしたから，今度はそれをひとつひとつ覚えていかないといけません．

そんな現状をなんとかしようと開発されたのがWindowsです．Windowsには，GUIを使った共通の操作性があり，誰にでもカンタンに操作できることを目標として作られたのです．ところが，そのプログラミングには，MS-DOSのプログラミング以上にたくさんのおまじないが必要なのです．

「環境がGUIならば，プログラミングもGUIになればいいのに……」こう思うのが，自然でしょう．GUIの先輩であるMacintoshには，HyperCard

BASICプログラミング用語集
カレントディレクトリ　ディレクトリをとくに指定しなかったとき，MS-DOSがファイルを検索するディレクトリのこと．

というモノがあります．これは，まさにGUIを使ってカンタンにプログラミングができるものです．「Windowsにも，こんなソフトがあったらいいなぁ」と思っていたら，ありました．Actor と ToolBook です．いずれも，Windowsのプログラミングを GUI でやっちゃおうというモノです．しかし，いくらGUIでプログラミングできるといっても，処理内容は専用の言語で書かないといけません．これがどうにも馴染めないものでした．また，細かいことをやろうとすると，どうしても足らない部分が出てきます．これは，「言語というよりはマクロだからしかたない」ですませるには，ちょっとくやしいところです．

そこに登場したのが，Windowsを作っているマイクロソフトの Visual-Basicです．正式には，Microsoft VisualBasic Programming System for Windowsといいます．その名のとおり，VisualBasicで使う言語はBASICです．実際には，Quick-BASICの仕様によく似てはいますが，Windows独自のプログラミングに合うように改良されているものです．もちろん，ウィンドウやダイアログは，GUIを使ってパーツをおいて作ることができます．それらの個々のパーツごとにBASICでコードを書いていけば，Windows用のアプリケーションができてしまうのです．実にカンタンでわかりやすいものです．ただ単に「起動すると，デザインしたウィンドウが表示されて楽しいな」というだけなら，BASICのコードすら書かなくてすみます．いざ，プログラミングをしだせば，QuickBASICが元になっているだけあって，いろいろな細かい処理もこなすことができます．

これならば，MicrosoftC と SDK を使って作るよりも512倍はカンタンにプログラミングを楽しむことができそうです．

Windowsプログラミングの考え方

VisualBasicでは，ウィンドウを作るだけならばBASICのコードを書くことすら必要ないといいました．とはいえ，やはりちゃんと動くプログラムを作りたいところです．Windowsのプログラミングをするには，やっぱりWindows独自のプログラミングの考え方をマスターする必要があります．マスターといっても，全部をおぼえる必要などありません．「へぇ．そんなもんかな」程度の気持ちになれればいいのです．あとは，習うより慣れろでVisualBasicをいじっていけば，おのずとわかってくるでしょう．

そこで，ここではWindowsプログラミングのためのごく基礎知識を仕入れておきましょう．

Windows独自のプログラミングの考え方というからには，Windowsがいままでの MS-DOS とどこが違うかを思い浮かべればよさそうです．すぐに思いつくのは，つぎの２つではないでしょうか．

1．GUIを使っている
2．マルチタスクである

「なぁんだ．あたりまえじゃないか」と思ったでしょうか．でも，そのあたりまえが大切なのです．では，１つ１つ考えていきましょう．

GUIを使っている

GUI（グラフィカルユーザーインターフェース）は，その名のとおり「グラフィックを使ったユーザーインターフェース」です．つまり，プログラムに何か入力するときも，プログラムから何か出力するときも，グラフィックを介するわけです．たとえば，ボタンのグラフィックをクリックしたり，ファイルメニューのグラフィックをダブルクリックします．これが入力です．キーボードから入力するときも，カーソルがグラフィックで表示されているところに文字が入ります．一方，出力もそれぞれ決まった位置にグラフィックで表示されたり，ウィンドウが出てきたりします．

この作業には，入出力に特定の関数が使われています．つまり，従来のBASICでやっていたように，

```
LOCATE 10, 5: INPUT A$:
PRINT " Hello ", A$
```

のような勝手なことはできないということです．もし，こんなことをしたら，ウィンドウのないところに文字を出してしまうことになりかねません．ウィンドウとは無関係にプログラムが動いてしまうのでは，何のためのGUIなのかわかりません．ですから，「プログラムの入出力は，VisualBasicで用意されている専用の関数を使いましょう」ということです．これらの関数を使えば入力文字列の長さによって画面が崩れるとかいう心配もありません．うまく使えるようになれば，こんな便利なことはないのです．

もっとも，VisualBasicの場合には，ウィンドウの外に勝手に文字を出すなんてことは簡単にはできません．ところが，Cなどを使うとこれができてしまうのです．ここでは，入出力の

BASIC プログラミング用語集
カレントドライブ　MS-DOSのキー入力待ちの状態で，プロンプトに表示されているドライブのこと．ディフォルトドライブとの呼ぶ．

関数は，Windows専用のものが用意されているのでそれを使おうというくらいに考えてください．

マルチタスクである

MS-DOSは，シングルユーザー／シングルタスクのオペレーティングシステムです．これは，同時に複数のプログラムは動作しないということです（常駐型プログラムというモノは同時に動いているように見えます）．これに対して，Windowsは同時にウィンドウ単位で複数のプログラムが動作します．いわゆるシングルユーザー／マルチタスクです．しかし，ここがクセモノです．もともとシングルタスクのMS-DOSの上に載っているWindowsがマルチタスクというのは，とってもおかしな話ではないですか．

そう，実はこのマルチタスク，実のところはかなり怪しいモノです．

通常，マルチタスクでも，CPUは1つです．したがって，動いているプロセス（動作しているプログラム）ごとに一定時間を割り振って細かく切り替えているのです．それでも，人間の目から見たら，同時に動いているように見えるわけです．この時間を決めてプロセスを切り替える手法をタイムスライスと呼んでいます．Windowsでは，メッセージと呼ばれるモノがWindows内部で飛び交っています．このメッセージがくると，そのプロセスの作業をCPUがやりはじめるのです．

しかし，Windowsではタイムスライスをしていません．というのも，1つのプロセスが処理をしはじめると，その処理が終るまでは中断することができないからです．ですから，メッセージはどんどんたまる一方です．これでは円滑にマルチタスクできません．そこで，CPUを占領し続けるような処理をしてはいけないわけです．

しかし，実際には，そこでディスクをアクセスして大量のデータを検索するとかいうこともあるでしょう．そういうときは，砂時計のアイコンを出して「処理に時間がかかっているよ」と知らせればいいのです．いずれにしても，あまり時間がかかるような処理は最初からしないようにすべきです．

さて，メッセージについてもう少しお話しましょう．Windowsはメッセージ駆動型といわれる方法をとっています．なんらかのメッセージが送られたら，それに対応する処理をするわけです．したがって，いままでのMS-DOSでのプログラミングのように，処理の流れをフローチャートに書いて追っていけるようなプログラミングはできません．処理はすべてメッセージに対応するカタチで書かなくてはならないのです．

これは，最初はとまどうのですが，慣れてしまえばなんてことはありません．実際にプログラムを書くときには，メッセージはウィンドウの上にあるパーツ（これをコントロールと呼んでいます）ごとに決まってきます．ですから，プログラムに必要なコントロールごとに設定すればいいわけです．

Windowsでのプログラミングの概念は，だいたいおわかりになったでしょうか? ここで，もう1つだけ注意すべき点をあげておきます．Windowsでは，動的にメモリを割り当てます．つまり，ある変数のアドレスは一定ではないということです．ですから，変数のアドレスを調べても意味がありません．もっとも，VisualBasicではそういう危険なことは最初からできないようになっています．

サンプルプログラミング1

さて，いつまでもゴタクを並べていてもしかたありませんから，そろそろVisualBasicを使ったプログラミングをしてみましょう．とはいえ，VisualBasicはいままでのプログラミング言語とはかなりかけ離れた環境です．その内容も盛りだくさんですから，ここではひと通りプログラミングに必要なメニューを開きながら，カンタンなプログラムを作ってみましょう．

ホントはインストールもしなくてはいけないのですが，インストールはとくに難しいことはないので，ここではすでにインストール済であることを前提とします．

ここで作るプログラムは，コマンドボタンをクリックすると，ビットマップデータを表示するというものです．

起動

VisualBasicのアイコンをダブルクリックして，プログラムを起動します．すると，ウィンドウが次々と表示されます．プログラミング環境というくらいですから，QuickBASICなどのように1つのウィンドウの中でなんでもできると思っていたら，VisualBasicはちょっと違うようです．

さて，画面の一番上にある横長なのがVisualBasicのメインウィンドウです．その下左にあるのがToolboxウィンドウ，右にあるのがProjectウィンドウといいます（写真1）．それぞれの機能は，使いながら説明します．

中央に「Form1」というタイトルの

BASICプログラミング用語集
キーワード　ステートメント・関数・演算子など，あらかじめ処理系形言語によって定義されている単語のこと．

ウィンドウがあります．このウィンドウが，これからプログラミングするウィンドウで，Formと呼んでいます．VisualBasicでは，各ウィンドウはこのFormという単位で考えます．

Formの大きさの変更

では，Form1の大きさを変えてみましょう．一般的なウィンドウのサイズを変更するのと同じように，ウィンドウの端をつまんでドラッグすると，ウィンドウの大きさを変えることができます．このプログラムでは，コマンドボタンとビットマップデータを表示する程度の大きさがあればいいわけです．だいたいの目星をつけて大きさを決定します．このサイズは後からでもカンタンに変更がききますから，あまり神経質になる必要はありません．

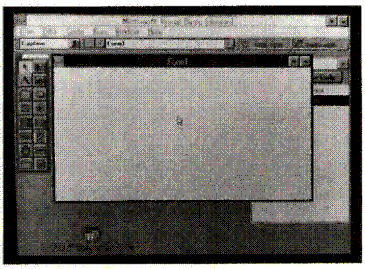


写真1　VisualBASICの起動画面

プロパティバーを操作する

さて，ここでメインウィンドウに注目してください．最下行にコンボボックスが2つあります．この行をプロパティバーと呼んでいます．

左側のコンボボックスを開くと，英単語がたくさん出てきます(写真2)．これはプロパティと呼ばれるものです．日本語でいえばさしずめ属性といったところです．ファイルなどの属性はアトリビュートですから，一般的にWindowsのプログラミングでは属性といわずにプロパティと呼ぶことにしています．

ここで，CtlNameプロパティを選択してみましょう．CtlNameとは，コントロール名のことです．コントロールとは，ウィンドウを作るパーツのことで，オブジェクトなどと呼ばれることもあります．いま，右側のコンボボックスには，Form1と表示されています．さきほどの左側のコンボボックスに出てきたプロパティは，Form1コントロールについてのものであるわけです．

ウィンドウ上のパーツ(コントロール)には，いろいろな属性(プロパティ)があります．たとえば，プログラムで扱うときの名前，色，大きさ，可視，不可視，フォントなどがあります．これは，コントロールによって違います．

次に，左側のコンボボックスからCaptionを選択してみましょう．右側にはやはりForm1と出ています．ここをクリックして文字列を消去してから，「TEST」と入力してみましょう(写真3)．Form1のタイトルは，先ほどまでForm1だったのが，TESTになりましたね．このように，コントロールの各プロパティの初期値はプロパティバー使って変更することができます．

プロパティバーの左側のコンボボックスのプロパティ一覧を見れば，だいたいどのようなものを変更できるかわかるでしょう．それぞれのプロパティについて，右側のコンボボックスの一覧を見れば，どのように変更できるかもわかります．

Pictureコントロールの配置

いよいよ，ウィンドウの中身を作っていきましょう．Toolboxウィンドウには，VisualBasicで使うことのできるコントロールの一覧がアイコンとして表示されています．各アイコンの意味は，図1のようになっています．

ここでは，まずピクチャーボックス

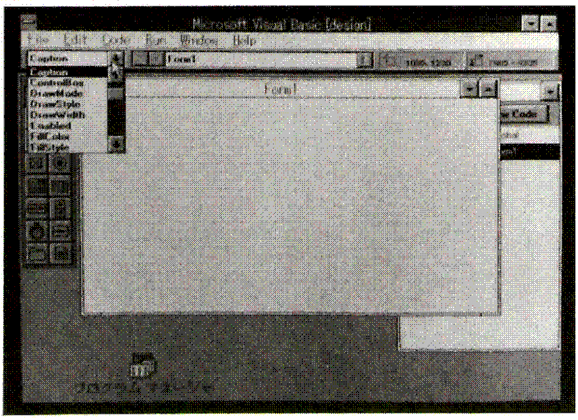


写真2　プロパティ一覧

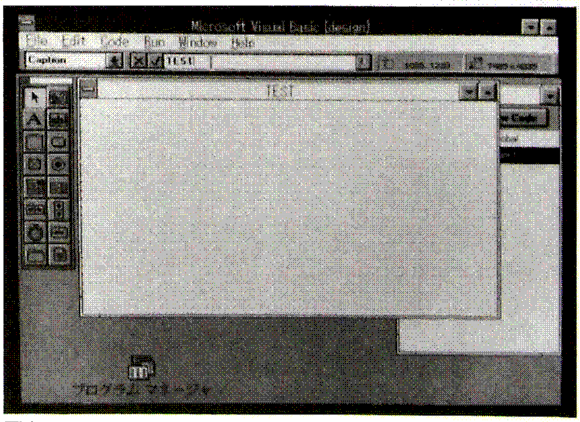


写真3　タイトルを変更する

BASIC プログラミング用語集

キャッシュ　RAMに比べデータの読み出し速度が遅いHDDなどで，キャッシュは読み出されたデータをRAMに蓄積し，そこから次の読み出しを行うようにして高速化する．

を選択します．Form1ウィンドウの上をドラッグすると，その2点を対角線とする長方形が表示されます．この長方形の上に，ビットマップデータを表示することができます．

このコントロール名はPicture1です．もし，もう1つPictureコントロールをおくと，Picture2になります．CtlNameがPicture1になっているか，プロパティバーで確認してください．

次に，左側のコンボボックスで，Pictureプロパティを選択してください．右側のコンボボックスをクリックすると，ここに表示するファイルを読み込むためのウィンドウが開きます（写真4）．ファイルを指定すると，もうそこには指定したビットマップデータが表示されてしまいます．ここでは，Windows付属のビットマップを指定してみました．FormのPictureコントロール上にビットマップデータが表示されます．

とりあえず実行

ここで，プログラムを実行してみましょう．え？ まだプログラミングしていない？ いいえ，そんなことはありません．すでにウィンドウを作ってビットマップデータを表示するプログラムを作っているではありませんか．何もおまじないのようなコードを書くだけがプログラミングではないのです．データとコードを1つのオブジェクトとしてとらえるのが，オブジェクト指向プログラミングです．データを用意し，それを表示するための準備は終っています．これぞ，オブジェクト指向プログラミングなのです．

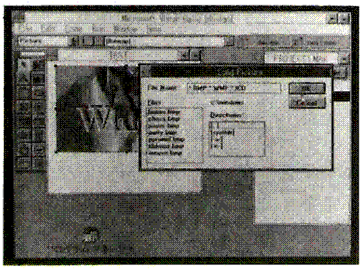
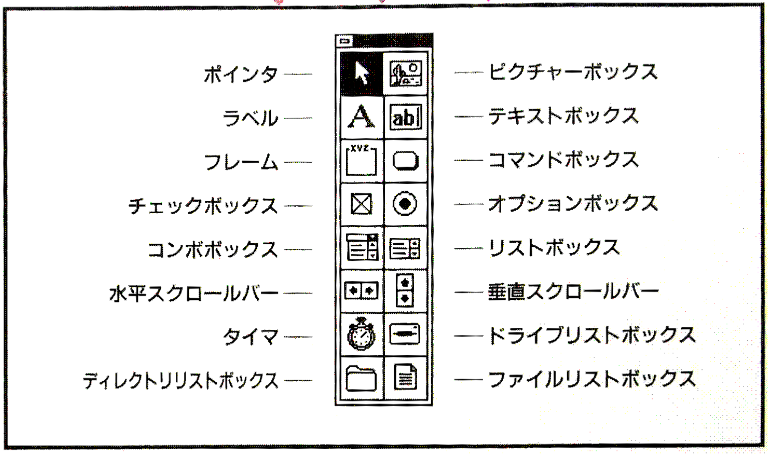


写真4　ファイルの読み込み

図1　アイコンの意味



Runメニューから Startを選んでください．まもなく先ほど作ったウィンドウが表示されます．ビットマップもきちんと表示されていますね．ウィンドウ左上のクローズボックスをダブルクリックすると，プログラムは終了します．

これで，すでにWindowsのプログラムができてしまったわけです．

Visibleプロパティの変更

しかし，これではあまりにもあっけないですね．もうちょっといじってみましょう．次に，さきほどのPictureコントロールをクリックします．ここで，プロパティバーからVisibleプロパティを選択します．いまは「True」になっているハズです．これを「False」にします．やりかたはわかりますね．コンボボックスをうまく使ってください（写真5）．

変更が終ったら，もう一度実行してみます．どうですか？ 今度はビットマップが表示されませんね．これは，ビットマップが見える「True」の状態から，見えない「False」の状態にプロパティを変更したからです．

Commandコントロールの追加

当初の仕様を実現するため，次のステップにいきましょう．

ここでは，Commandボタンを追加します．ToolboxからコマンドボタンJを選択し，Pictureコントロールをおいたときと同じようにForm上にCommandボタンを配置します．コントロール名はCommand1になっています．

プロパティバーを操作して，Captionプロパティを「Push Here」に変更します．これで，ボタン上の文字列も「Push Here」になりました．

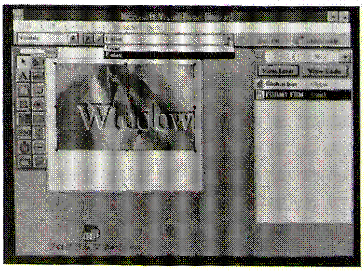


写真5　Visibleプロパティの変更

BASICプログラミング用語集
行番号　BASICなどのプログラムの行の先頭に記される番号．行をメモリに格納したりディスクにセーブするときの格納順序，分岐処理の分岐先の指定にも使われる．

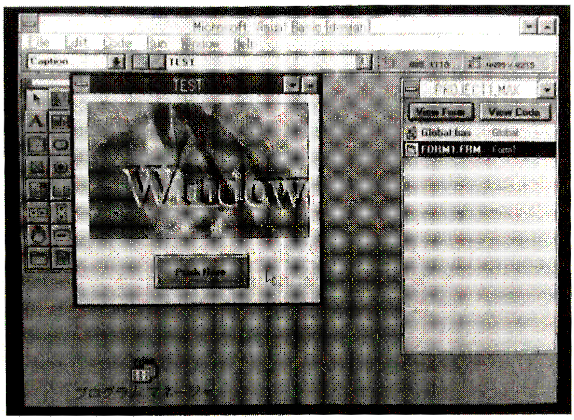


写真 6　Command ボタンを追加する

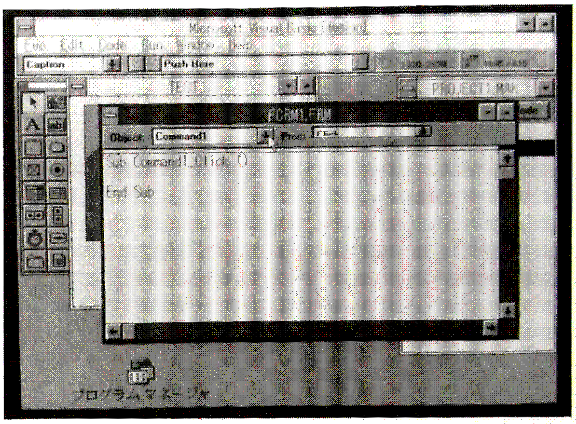


写真 7　関数の枠組み

これで，このプログラムに必要なコントロールはすべて作りました（写真6）．ためしに，Startしてみましょう．ちゃんとウィンドウが表示されました．しかし，ボタンをクリックしても何も起きません．

コードの組み込み

これは，ボタンがクリックされたら何をすればいいかという指示を，どこにも書いていないからです．この部分は，QuickBASICに似た文法のBasicでコードを書いてやらないといけません．VisualBasicでは，見かけの部分はコードで記述しなくてもプログラミングできますが，実際の処理内容はコードで書く必要があるわけです．

ここでは，「Command1コントロールがクリックされたら，Picture1コントロールのVisualプロパティを"True"にする」という処理が必要になります．

では，Form1上のCommand1コントロールをダブルクリックしてください．新しいウィンドウが開きました(写真7)．よく見ると，すでに何かコードが書いてあります．「Sub Command1_Click ()」と「End Sub」とあります．これは，QuickBASICのプロシージャの定義と同じです．つまりこれが中身のない1つの関数なのです．私達は，この中にやりたい処理を書いていけばいいわけです．

さて，このウィンドウをよく見ていきましょう．タイトルは「FORM1.FRM」です．Form1についてのコードを書くところなわけです．この名前でファイルにセーブすることもできます．その下には，2つのコンボボックスがあります．左には「Object」とあります．このコンボボックスを開くと，いま配置してあるコントロールの一覧が表示されます(写真8)．右側の「Proc」コンボボックスを開くと，プロシージャの一覧が表示されます(写真9)．このプロシージャは，それぞれのコントロールにどのようなことがされたかによって，名前がつけてあります．さきほどの「Command1_Click ()」は，Command1コントロールがクリックされたときの処理を書くところです．

ここでは，最初に表示されていた「Command1_Click ()」の中身を書きましょう．

```
Picture1.Visible = -1
```

この1行で終りです．Picture1のVisibleを「-1」にするという意味です．[コントロール名].[プロパティ名]で記述していることをおぼえてください．ここで，「-1」は「True」を意味します．ちなみに，「False」は「0」です．

実行

これでプログラミングはすべて終了しました．実行してみましょう．今度は，コマンドボタンをクリックすると，ビットマップが表示されました．

EXEファイルの作成

プログラムが動作することを確認し

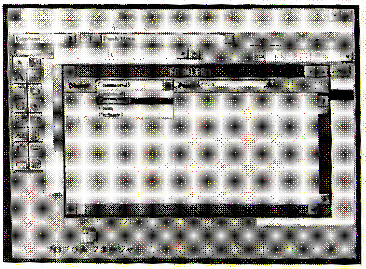


写真 8　コントロール一覧

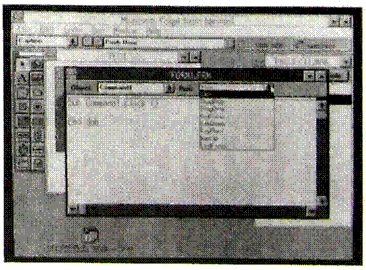


写真 9　プロシージャ一覧

BASIC プログラミング用語集
クリック　マウスのボタンを押すこと．マウスのボタンを連続で2回押すことをダブルクリックという．

リスト1　FORM1.TXT

```
Sub Command1_Click ()
    Picture1.Visible = -1
End Sub
```

たら，今度はEXEファイルを作ってみましょう．いままではインタープリタで動作していましたから，Visual-Basic環境からでないとプログラムが動作しないのです．

メインメニューの「File」から，「Make EXE File」を選択してください．あっという間に作業は終って「TEST.EXE」ができます．このファイルは実行形式とはいえ，ランタイムライブラリが必要です．これは，「vbrun100.dll」というDLL形式の270KBほどの大きさのファイルです．スタンドアローンで動いてくれないのは少々残念です．起動時にこのDLLを読み込みますから多少時間がかかりますが，その後の実行速度はそれほど遅いものではありません．

このように，VisualBasicでは，MS-DOS用のプログラムを作るより手軽にWindowsプログラミングを楽しむことができます．しかし，1行程度のプログラムでは，Windowsプログラミングの奥深い楽しみや可能性はわからないかもしれません．だいたい，これでは「どこがBasicなんだ」と思ってしまいます．Windowsプログラムの特徴であるメニューすら使いません．

そこで，次に3つほどサンプルプログラムを用意しました．VisualBasicの基本操作は理解していただけたと思いますので，今度は要点だけを説明します．

サンプルプログラミング2

このプログラムでは，入力された文字列を編集して出力してみましょう．次の手順でコントロールを配置します．

①Form1のサイズを変更します．
②Labelコントロールを配置します．
③Label1.Captionプロパティを「Plese type your name.」に変更します．
④Textコントロールを配置します．この Text1コントロールに文字列が入力されます．
⑤Text1.Textプロパティの内容をBSキーで消去します（ここまでのForm1は写真10）
⑥Text1コントロールにコードを組み込みます．Proc:一覧を確認し，「Text1_KeyPress」を選択します．このプロシージャは1文字入力されるごとに呼び出されます．引数「KeyAscii」は入力された文字のアスキーコードです．Text1.TextはText1コントロールに入力された文字列です．リターンキーが押されたら，Label1のCaptionプロパティを変更するようにしましょう．リスト2のコードを入力します．

これはまさにQuickBASICの文法です．文字列操作がカンタンな，BASICならではの書き方が可能です．

⑦Startします．リターンを押したときにLabel1の文字列が変わることがわかるでしょう（写真11）．

このように，VisualBasicでは外部から加えられた操作に対応して，コントロールのプロパティを変更することによってプログラムが動作しています．

⑧これだけでは寂しいので，色をつけてみましょう．メインウィンドウから「Window」を選択し，そ

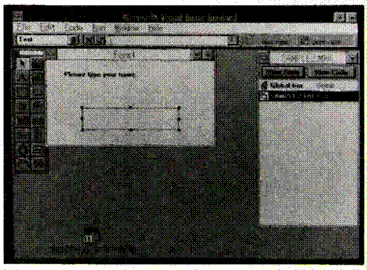


写真10　Text1.Textプロパティの内容をBSキーで消去したところ

リスト2

```
Sub Text1_KeyPress (KeyAscii As Integer)
    If KeyAscii = &HD Then
        Label1.caption = "Hello, " + text1.text + "-san."
    End If
End Sub
```


BASICプログラミング用語集
構造化　オランダのE.W.ダイクストラが提唱したプログラミングの手法．プログラム中でGOTO文の使用を避け，処理内容に合わせてブロックに分けて作成する方法

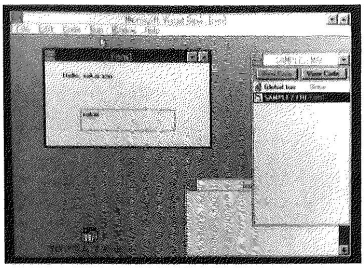


写真11　Label1の文字列が変わる

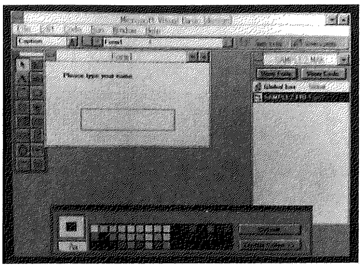


写真12　ColorPaletteを選択する

の中から「ColorPalette」を選択します．下の方にカラフルなウィンドウが出てきます(写真12)．このウィンドウを使えば，各コントロールの色を変更できます．

色はプロパティですから，プロパティバーのBackColorとForeColorでも変更できます．もちろん，コード中に直接書いて変更することもできます．

サンプルプログラミング3

いままでのプログラムでは，必ず外部から加えられた操作によって処理をしていました．ここでは，タイマを使った面白いプログラムを紹介します．

タイマは，一定時間ごとになんらかの処理をするときに有効です．たとえば，ウィンドウの中をボールが動き回っているプログラムを作ろうとします．しかし，ずっとボールが動き回っている処理をしていたら，ほかの処理ができなくなってしまいます．Windowsでは，1つのプログラムがCPUを占領してしまうとマルチタスクを実現できないことは，前にも書きました．

このような処理は，たとえば100分の1秒ごとにそのルーチンが呼ばれるようにすれば，それ以外の時間にはほかの処理をすることができます．それでも，人間からの見た目には，連続して処理が行われているように見えるのです．これを実現するのがタイマです．

ここでは，タイマを使ってボールがウィンドウの中を動き回るプログラムを作ります．

①Formのサイズを変更します．

②Pictureコントロールを配置します．

③Picture1のプロパティを変更します．ここで，あらかじめ用意したボールのビットマップを，Pictureプロパティで選択します．ビットマップのサイズをWidthとHeightプロパティで変更します．ここでは，32×32ドットなので，32×16=512にしました．また，BorderStyleをNoneにします．

④Picture1_Clickプロシージャに

Unload Form1

とコーディングします．つまり，このビットマップをクリックしたら終了することにするわけです．

⑤Timer1コントロールを配置します．場所はどこでもかまいません(ここまでのForm1は写真13)．

⑥Timer1のIntervalプロパティを100に指定します．この単位は1000分の1秒単位ですから，10分の1秒ごとに割り込みがかかるということになります．

⑦Timer1のProc:一覧を確認してください．Timerしかありません．

⑧Timer1_Timerにリスト3に示すコードを組み込みます．ここで，ビットマップがウィンドウの端で跳ね返るたびに，sxとsyは符号が逆転する必要があります．したがって，この2つの変数はグローバル変数でなくてはいけません．

⑨ProjectウィンドウからGlobal.basを選択します．ここで開いた

リスト3

```
Sub Picture1_Click ()
    Unload Form1
End Sub

Sub Timer1_Timer ()
    Picture1.Visible = 0
    Picture1.Left = Picture1.Left + sx
    Picture1.Top = Picture1.Top + sy
    Picture1.Visible = -1
    If Picture1.Left + Picture1.Width > Form1.Width Or Picture1.Left < 0 Then
        sx = -sx
    End If
    If Picture1.Top + Picture1.Height > Form1.Height Or Picture1.Top < 0 Then
        sy = -sy
    End If
End Sub

Sub Form_Load ()
    sx = 100
    sy = 100
End Sub
```

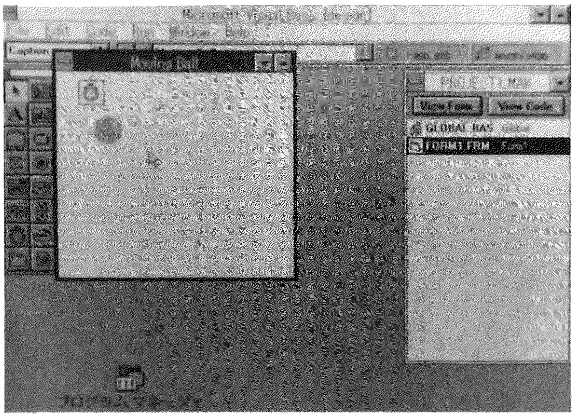


写真13　Timer1コントロールを配置したところ

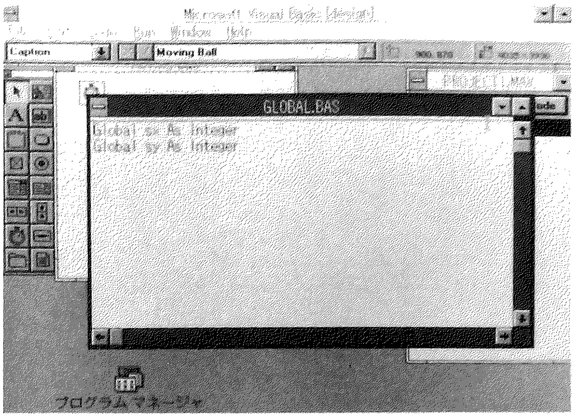


写真14　グローバル変数を記述する

ウィンドウは，グローバル変数を書くところです．ここでは，

```
Global sx as Integer
Global sy as Integer
```

のように書きます(写真14)．ここでは宣言しかできません．

⑩Form_Loadプロシージャを選び，sxとsyの初期値を入力します．ここでは100にしました．Form_Load()は，Form1がロードされ画面に表示されると同時に実行されるプロシージャです．

⑪Startします．ほかのウィンドウで別の処理をしていても，ボールはウィンドウ内を動き回っています(写真15)．これは，タイマーを使っているからです．Timer1.Intervalプロパティを変更すると，この処理にかかる時間が変わります．いろいろと値を変えてみてください．

このプログラムで面白いのは，ビットマップをカンタンに移動させるアイデアです．Pictureコントロールには，LeftとTopのプロパティがあります．この値を変更すると，表示される位置も変わるのです．

マルチタスクを実感できるのも楽しいところですが，Windowsではタイマーを合計16個までしか使うことができません．調子にのって使いすぎないように気を付けましょう．

サンプルプログラミング4

ここまでのサンプルでは，Windowsのプログラムなのにメニューはありませんでした．ここでは，メニューのあるプログラムを作ってみましょう．少しは役に立つ(かもしれない)プログラムということで，ビットマップを手軽に見れるViewerを作りましょう．

①Formのサイズを変更します．

②メインメニューの「Window」を開き「MenuDesign」を選択すると，Menu Designウィンドウが現れます(写真16)．

③メニューの名前を決めます．Caption:のところに「&File」と入力します．&はALT(一部機種ではGRPHなど別のキーになっています)と同時に押したときに開くキーの前に書きます．ここでは，[ALT]+[F]キーでこのメニューが開くことを意味します．

④コントロール名を決めます．CtlName:のところに入力します．プロシージャの頭につくものやプロパティの頭につくものとと同じモノです．ほかのコントロールと重ならないようにします．ここでは，「file」にしました．

⑤「Next」ボタンをクリックすると，次のメニューアイテムを入力できます．3.4.と同様にして，「New File」「Open File」「Save File」「Exit」を定義します．ここで，「New File」と「Save File」はこのプログラムでは意味があり

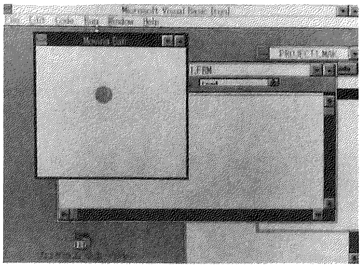


写真15　ボールはウィンドウ内を動きまわる

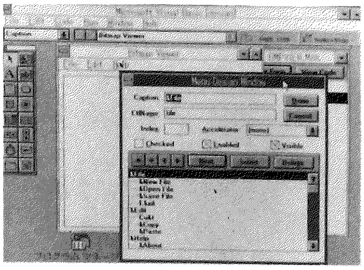


写真16　Menu Design Window

BASIC プログラミング用語集
コンパイラ　ソースファイルを機械語または，オブジェクトと呼ばれるコードに翻訳する．さらにリンカで各オブジェクトをリンクして実行可能なプログラムを作成する．

ません．そこで，この2つを定義するときには,「Enable」のチェックを解除します．

⑥メニュー階層を決定します．この4つのメニューアイテムは,「File」が選択されたときにドロップダウンメニューとして出てくるためのものです．そこで，これらの4つについては，それぞれ「→」ボタンをクリックして右に一段動かします．

⑦同様にして,「Edit」メニューから「Cut」「Copy」「Paste」の3つからなるドロップダウンメニューが開かれるように定義します．

⑧同様にして,「Help」メニューから「About」だけのドロップダウンメニューが開かれるように定義します．

⑨定義が終了したら,「Done」を入力します．コントロール名がおかしくなければ, Form1にメニューが追加されているのが見えます．

⑩「File」メニューをクリックします．ドロップダウンメニューが表示されたら,「Open File」をクリックします．このメニューに対応したコードを書くためのウィンドウが開かれます．ここでは，ファイルを選択するためにウィンドウを開くことにします．このウィンドウはまだ作っていませんが,「Form2」というコントロールであるとしましょう．「openfile_Click()」には,「Form2.Show」という一文を書きます（写真17）．

⑪同様にして,「exitprog_Click()」には,「Unload Form1」「about

リスト4

```
Sub about_Click ()
    Form3.Show
End Sub

Sub openfile_Click ()
    Form2.Show
End Sub

Sub exitprog_Click ()
    Unload Form1
End Sub

Sub cut_Click ()
    On Error Resume Next
    Clipboard.Clear ' Clear Clipboard.
    Clipboard.SetData Picture1.Picture, 2
    If Err Then
        Msg$ = "Could not find the .BMP file."
        MsgBox Msg$ ' Display error message.
        Exit Sub
    End If
    Picture1.Picture = LoadPicture()
End Sub

Sub copy_Click ()
    On Error Resume Next
    Clipboard.Clear ' Clear Clipboard.
    Clipboard.SetData Picture1.Picture, 2
    If Err Then
        Msg$ = "Could not find the .BMP file."
        MsgBox Msg$ ' Display error message.
        Exit Sub
    End If
End Sub

Sub paste_Click ()
    Picture1.Picture = Clipboard.GetData()
End Sub

Sub Form_Unload (Cancel As Integer)
    Unload Form3
    Unload Form2
End Sub
```

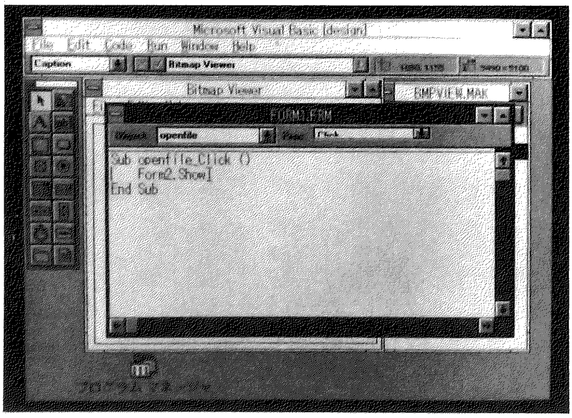


写真17　File メニューの Open file の内容を記述

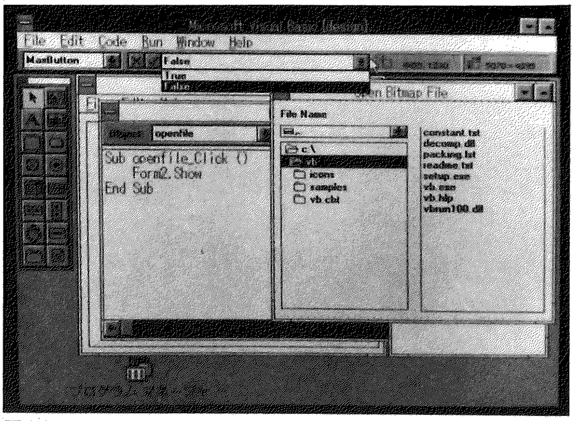


写真18　Form2の定義

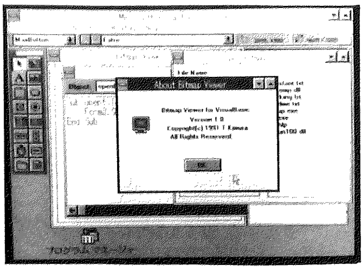


写真19　Form3の定義

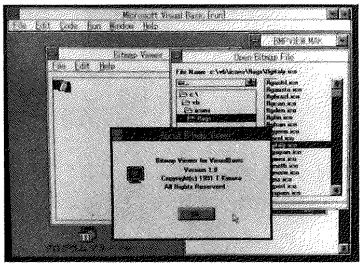


写真20　プログラムの実行

_Click()」には，「Form3.Show」とコーディングします．Form3はAboutダイアログボックスです．

⑫ Form2を定義します．メインメニューから，「File」をメニューを開き「New Form」を選択します．Form2があらわれます．ファイルを選択するウィンドウですから，「File Name」という文字列を表示するラベル，実際のファイル名を表示するラベル，ドライブリストボックス，ディレクトリリストボックス，ファイルリストボックスの各コントロールを配置します．プロパティも適切に変更します．ここで，Form2コントロールのMaxButtonとMinButtonのプロパティをFalseにしておきましょう（写真18）．

⑬ Form2へコードを組み込みます．リスト5がその内容です．

⑭ Form3を定義します．⑫と同様の手順で行います．Form3コントロールのCaptionとMax-ButtonとMinButtonのプロパティをFalseにすることをお忘れなく（写真19）．

⑮ Form3へコードを組み込みます．ボタンをクリックされたら「Un-load Form3」が実行されるようにします．

⑯ Editメニューの各項目について，コードを組み込みます．ここでは，クリップボードとのデータのやり取りをします．リスト6がその内容です．Clipboard.SetDataとClipboard.GetDataというプロシージャを呼び出すと，カンタンにクリップボードとのやりとりができます．

⑰ Form1_Unload()に，「Unload Form3」と「Unload Form2」の2行を書きます．これは，Form1が終了したときには，Form1の下請けである2つのウィンドウも閉じるようにするためです．

⑱ Startします．このプログラムでは，BMPとICOの2つファイル形式を読み込んで表示することができますが，クリップボードとデータのやり取りができるのはBMP形式のファイルのみです（写真20）．

以上，ちょっと急いでVisualBasicのサンプルプログラミングを紹介しました．舌足らずになった部分もあると思いますが，VisualBasicの環境をいじったり，コードの内容をよく見れば，いろいろなプログラミングへのヒントがあると思います．みなさんのプログラミングのご参考になれば幸いです．

リスト5

```
Sub Drive1_Change ()
    On Error GoTo TRAP:
    File1.Path = Drive1.Drive
    Dir1.Path = Drive1.Drive
    Label2.Caption = Dir1.Path + File1.Filename
    Exit Sub
TRAP:
    MsgBox "Drive not ready"
    Resume RES:
RES:
    On Error GoTo 0
End Sub

Sub Dir1_Change ()
    File1.Path = Dir1.Path
    Label2.Caption = Dir1.Path + File1.Filename
End Sub

Sub File1_Click ()
    Dim sep As String
    Dim BMPFilename As String

    If Right$(Dir1.Path, 1) = "¥" Then
        sep = ""
    Else
        sep = "¥"
    End If
    BMPFilename = Dir1.Path + sep + File1.Filename
    Label2.Caption = BMPFilename
    Form1.Picture1.Picture = LoadPicture(BMPFilename)
End Sub

Sub Form_Load ()
    File1.Pattern = "*.bmp;*.ico"
End Sub
```

BASICプログラミング用語集
サブルーチン　比較的小ステップの命令で構成され，主プログラムから複数回呼び出され，一定の処理を行う命令群のこと．実行後は主プログラムに制御が戻る．

その他の環境とプログラミング

このように，Windowsで通常必要と思われるプログラミングは，すべてVisualBasicでできてしまいます．しかも，従来のC+SDKという環境に比べて，格段に使いやすく，格段に安い値段で実現できるのです．

このほかにも，VisualBasicにはたくさんの機能があります．

Windowsではもはやあたりまえの強力なヘルプ機能をうまく使えば，紙のマニュアルよりも有効にプログラミングのための情報を仕入れることができます(写真21)．

VisualBasic自身で書かれたチュートリアルを使えば，VisualBasicの環境からプログラミングまで，楽しく学習できます(写真22)．ただし，これはVGAのレゾリューションを前提に作られていますから，PC-9801のノーマルレゾリューションでは正常に動作しません．

いずれも強力なものですが，現在のところVisualBasicは日本語バージョンが発売されていません．したがって，ある程度英語がわからないとせっかくの強力な環境もチンプンカンプンです．

また，VisualBasicではDDE(ダイナミック・データ・エクスチェンジ)など，Windowsのもつ強力な機能を使うこともできます．使い込めば使い込むほど，その奥の深さに驚きます．

さらに，数百種類のアイコンが付属してきます．これを，サンプルプログラムのアイコンエディタで編集して，独自のアイコンとすることもできます．サンプルプログラムには，ほかに電卓とカードファイルが入っています．

マニュアルは，プログラマーズガイドとランゲージリファレンスの2冊が付属しています．

リスト6

```
Sub Command1_Click ()
     Unload Form3
End Sub
```

VisualBasicのメリット

ユーザーインターフェース部分はGUIでデザインし，その処理内容をBASICで記述できるVisualBasicは，まさに新世代のオブジェクト指向言語だといえるでしょう．しかも，発売されたばかりのQuickC for Windowsと組み合わせれば，処理内容はCを使うというワザも使えるのです．

いままで，Windowsのプログラミングには膨大な労力と時間が必要でした．プログラムの製作課程で仕様が変更されたりすると，ほとんどパニック状態になるような環境だったのです．しかし，VisualBasicを使えば，Windowsプログラムのプロトタイプもカンタンに作ることができます．これは，ソフトハウスにとって大変なメリットでしょう．

しかし，VisualBasicの一番のメリットは，誰もが手軽にWindowsプログラミングができるということでしょう．そのために選ばれたのが，Basicという文字どおりベーシックなな言語だったわけです．Basicインタープリタの特徴である手軽さ，文字列操作の容易さに加え，QuickBASICで培われた構造化プログラミングを取り入れ，拡張されたVisualBasic．これからも，Basicはいっそう洗練され，高機能になって私達の前に現れることでしょう．

最後に，VisualBasicの1日も早い日本語版の登場に期待して，この稿を閉じたいと思います．

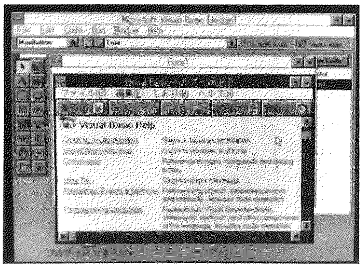


写真21　ヘルプ画面

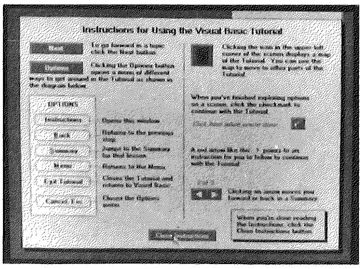


写真22　チュートリアル

BASIC プログラミング用語集
ショートサーキット　条件分岐などで，すべての論理式を評価するのではなく，前方から評価し，論理式の値が決まった時点で評価を決める方法

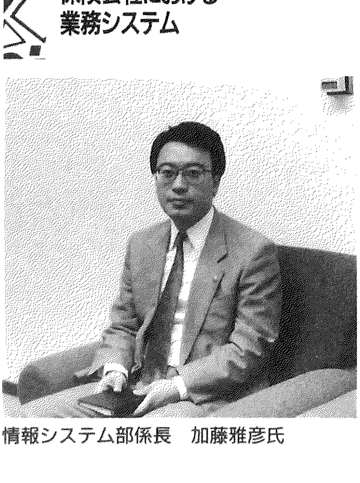
Interview

企業に生きるBASIC
50万ステップにおよぶBASICシステム

明治生命保険相互会社情報システム部係長
加藤雅彦氏 ―――に聞く

早くから社内のOA化を進めた企業には，長年に渡って築き上げてきた膨大なBASICプログラムの資産を現在も維持，活用しているものが少なくない．明治生命保険相互会社もこのような企業の1つである．ここでは，同社情報システム部係長である加藤雅彦氏にBASICによる業務システムの開発，保守，今後の展望などについて語ってもらった．

保険会社における業務システム


情報システム部係長　加藤雅彦氏

当社では，外交員の行う通常の業務を支援するためにパソコンを使っています．まず，設計書(見積書)の作成プログラムがあります．保険の契約時には，月々いくら払うとどれくらいの保障があり，最終的な配当がいくらになるかということを提示しなければなりません．パソコンを導入する以前には，分厚い配当表，保険料率表などを見て保険の種類を捜し出し，性別の欄を見て，年齢の範囲を見て……，というようなことをして金額を計算していました．このような方法だと慣れた人間でも1枚の設計書を作るのに30分くらいはかかってしまいます．これをパソコンを利用して簡単に作成できるようにしました．

次に保険会社では，お客様にいろいろな話題や情報を提供する「ツール」と呼ばれる情報提供物を数多く作成しますが，これらのツールをお客様のデータから選択して自動的に作成するプログラムがあります．

あと，顧客管理プログラムがあります．保険会社が管理するのは契約したお客様ですが，外交員が管理したいお客様というのは契約を締結していない，これからお客様になってくれそうな人です．この場合，まず会って話をして名前をなんとか聞き出し，誕生日などは話のおりに聞き出し……，ということになりますから不確定情報が多くなります．さらに，趣味だとか，どういう話題を出せばなかよくなれるかというような情報を集めて管理しなければいけないので，外交員自身が個別にお客様を管理できるようなデータ管理システムが必要になるわけです．

BASICを開発言語に選択

昭和58年に支社の事務効率を上げるためにIBMの5550を導入し，59年には営業所に5550の互換機を導入しました．そこで，主要な設計書，ツール類，データ管理システム，システム

機器管理システムを作って現地展開したわけです.

このときIBMの5550で使用可能だった言語が，　BASICとMS-Pascalでした. 最初は両方の言語で開発していたんですが, Pascalはとても使いづらかったのでBASICを採用することにしました. BASICには機種依存しているような命令も全部揃っていたのに, Pascalにはなにもなかったというのが BASICを選んだおもな理由です.

BASICプログラムのメンテナンス

最初のBASICは, IBMに特化したものだったので互換機用にBASCOM 5.35を購入したのですが，このBASCOMにはIBMのBASICにある機種依存した命令がなかったため，その部分をアセンブラで書き起こしました. こうしてシステムを完成させたわけです. この時点でもう10数万ステップという規模になり，1年から2年で20万ステップほどになったわけです.

毎年数万ステップずつ増えてくると，システムの保守がとても難しくなります. そこで少しでも保守を楽にするために，いろいろなルールを作っています. たとえば汎用のルーチンで使う変数は，必ず特定の文字から始まる変数名を使うとか，行番号ごとに汎用の処理を決めて，ここではこういうインターフェースを呼んで処理しなければいけないとかいうものです.

しかし，すぐに限界が見えてきました. そこでもう一度検討して，大規模開発に必要な構造化設計をサポートする言語で，さらに機種依存しないプログラムを作成できることが必要だと考えました. このときほんとうはC言語に変更したいと思ったのですが，それまで使っていたプログラマにCの経験がなく，しかも目の前には山のようなBASICのソースがあるわけですから，一からCで書き直すというわけにはいかなかったわけです. しかたなく，BASICの命令のうち機種依存しそうな命令をいっさい使わないようにして，その代替のルーチンだけをCで記述し，アセンブラのルーチンも廃しました. この頃のBASCOMは，マルチランゲッジをサポートしていなかったので，苦心惨憺したあげくにTurbo Cでなんとか BASCOMとリンクできるオブジェクトを作ったのですが，標準ライブラリを使うことができなかったので，アプリケーションレベルの記述にはBASICを利用するしかありませんでした.

こうして少しずつCのソースが増えてきましたが，全ソースの1割にもなりません. ほとんどは元の構造化されていないBASICのままです. この部分を解決するためにQuickBASICを採用しました. ほとんどのソースは，そのままでQuickBASICでコンパイルできたので，行番号のカットやGOSUBしていたところをCALLに置き換えるなどの処理をするコンバータを作成し，いったんこのコンバータにソースを通し，さらに細部を修正するという方法で移植を行いました. 実際の作業は今年の4月から始めて12月までに50万ステップほどあるソースの6割までを移行して，残り4割を来年の2月までに行うつもりです. 基本的にはコンバータで移行は終ってますから，あとの作業の大部分はテストです.

また，現在ではCなどで作成していた機種依存している部分はメモリに常駐する50KBほどのライブラリの形にして，EXEファイルにリンクするのはやめています. この部分だけで機種の差を吸収できればいいのですが，そううまくはいかないようです. 当社では3年ごとにパソコンのリプレイスが行われるのですが，これをいかに乗り切るかが現在の課題です.

将来はWindowsへ

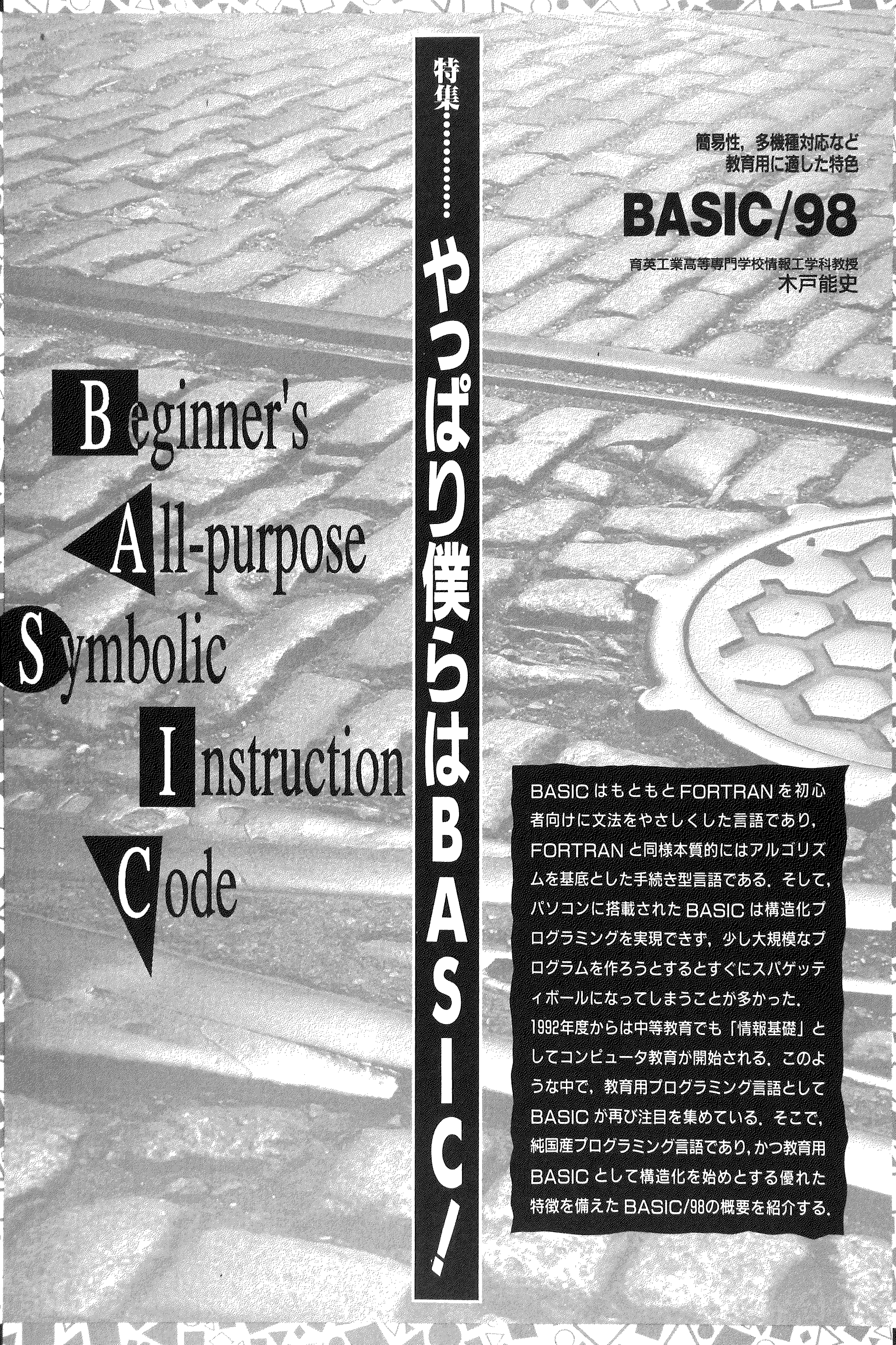
パソコンベースで見る限りでは，Windowsが主流になるだろうと考えているので，いまの形態をWindows上にもっていけば，そんなに苦労しなくても乗り越えられるんじゃないかと期待しています.

Windowsについては，バージョン1.03，2.0の頃から開発ツールなども含めて検討していました. OS/2についても検討したのですが，コストの問題などから見送っています. 環境を移行する場合には，必ず順次移行する必要があります. いっぺんにすべてのプログラムを移植することなどできませんから，いまのプログラムを使いながら新しい環境が動かなければならないわけです. WindowsならばDOSアプリケーションを動かせますから，問題なく移行できると考えています. もしもUNIXなどに移行するとしたら，いまのプログラムをどうやって動かすのかという問題が出てくるわけです.

Windowsもバージョン3.0が出て，VisualBASICも出てきたし，Cに移行しなくても今のシステムを順次移植できるのではないかと思います. また，並行してBASCOM 7.1についても評価を進めているところです.

そろそろBASICは限界かなと思っていたところにVisualBASICが出てきたし，たまたまわれわれはWindowsを指向していたので，ラッキーだったのかなというところですね.

BASIC プログラミング用語集
スタック　データを一時的に退避させること，もしくはそのための記憶領域を指す.



特集……

やっぱり僕らはBASIC！

簡易性，多機種対応など
教育用に適した特色

BASIC/98

育英工業高等専門学校情報工学科教授
木戸能史

Beginner's
All-purpose
Symbolic
Instruction
Code

BASICはもともとFORTRANを初心者向けに文法をやさしくした言語であり，FORTRANと同様本質的にはアルゴリズムを基底とした手続き型言語である．そして，パソコンに搭載されたBASICは構造化プログラミングを実現できず，少し大規模なプログラムを作ろうとするとすぐにスパゲッティボールになってしまうことが多かった．1992年度からは中等教育でも「情報基礎」としてコンピュータ教育が開始される．このような中で，教育用プログラミング言語としてBASICが再び注目を集めている．そこで，純国産プログラミング言語であり，かつ教育用BASICとして構造化を始めとする優れた特徴を備えたBASIC/98の概要を紹介する．

BASICプログラミング 教育の混乱

1980年にPC-8001が登場して10年余が経過した．わが国で普及した初のこのパソコンにバンドルしていたプログラミング言語がN-BASICであった．これは後に，N88-BASICに仕様拡張されたが，現在に至るまで基本仕様は変更なく使われている．パソコンを購入した人が電源を入れると，直ちにプログラムが使える，コンピュータを動かせるという感激を味わった．それは，それまでのコンピュータの概念を覆すたいへんなできごとであり，DOSなどのめんどうな知識と手続きなしに，直ちに計算ができ，画が描け，ゲームで遊べるという画期的なものであった．

しかし，このときマシンとともに無償で提供されたこのBASICが，後に大きな禍根を残すことになった（というのはいい過ぎか……）．パソコンが200万台以上も普及した今日までに，プログラムを作ろうと思いついた人々が，BASICというその名称もやさしそうな，簡単に学べるのではという，このプログラミング言語に挑戦し，そのほとんどの人が挫折していった歴史がある．これは，別にN88-BASICだけの責任ではなく，BASICはもともとFORTRANという技術系の言語を，初心者向けに文法をやさしくした言語であり，文字列処理と描画処理も簡単に操作できる言語として登場した．しかし，BASICはFORTRANと同様に，本質的にはアルゴリズムを基底とした手続き型言語であるところから，その意味での難しさに変わりがなかった．

筆者も長くBASICの教育に携わってきて感じてきたことは，アルゴリズムを表現するのにより適切な構文がBASICに用意できないものなのか，未だに大学などのプログラミング言語教育で，Pascalが実用言語でないにもかかわらず，主要な位置を占めているのはそこに理由があるのではと考えていた．また，卒業研究などで数百ステップ以上の多少まとまったアプリケーションを開発させようとすると，BASICはあっというまにスパゲティボールを作ってくれるという経験を何回もしている．情報を専門とする大学の先生からいわせれば，「決してBASICは教えてくれるな」とまでいわれる始末である．

このような経過の中で，1992年度からは中等教育でも「情報基礎」の必修を始めとして，各教科でコンピュータの利用が積極化する方向にあり，その中で再びあのBASICが教えられるかと思うと，複雑な気持ちにならざるを得ない．

構造化BASICの登場

ところで1987年ころより，構造化BASIC言語ソフトが次々と市販されるようになり，今日QuickBASICがその代表的なものとなっているが，同時期の1987年に開発され，現在日本SE株式会社グループの株式会社ケーエスピー（KSP）が開発販売している数少ない純国産プログラミング言語「BASIC/98」は，次の理由から教育用にもっとも適したプログラミング言語である．

①N88-BASICの仕様を99%包含しつつ，必要な構造化構文を十分に用意している．従来のBASICに慣れている先生も移行しやすい．

②統合環境ではないが，教育分野では逆に簡単なインタプリタで十分である．統合環境と編集操作に慣れるのは，初心者にはけっこうたいへんであり，マウスなど余分な装置が必要になることもある．

③インタプリタではあるが，実行速度の点からインクリメンタルコンパイル方式を採用し，実質的なコンパイル実行となっているので実行速度が速い．

④エラーメッセージが日本語，英語，コードの3種から選択できる．とくに，日本語があるので初心者にとっては理解しやすい．

⑤教育現場では，学校や学習者の両者を含めて考えると，1社のパソコンで占められていることはなく，複数のメーカー，多数の機種で稼働することが重要である．その意味で互換性が高いことが重要である．BASIC/98は，NEC，松下，富士通，日立，シャープ，沖，IBMなどの多機種に対応している．NEC以外の機種でN88-BASICが動き，なお構造化構文を使える．

⑥高等教育機関（大学，短大，高専，専修など）において，教育のみならずシステム開発までの拡張性をもつ．ただし，メモリ制限からはC言語などのほうが有利である．

⑦民間でも，意外とBASICが開発言語として用いられているという最近の報告もあり，実行形式ファイルがライブラリと切り放して単独に実行可能である．著作権の関係でライブラリや実行ドライバな

スタティック配列　プログラムをコンパイルするとき，メモリが割り当てられる配列のこと．プログラム実行中にアドレスが変更されることはない．

どを搭載してリンク形式での販売は難しい．

以上の理由から，教育用のみならず開発用言語としても，十分に対応できることがわかる．一般・教育用を対象としたBASIC/98Fast版以外に，開発用としてBASIC/98Pro(EMS仕様)も提供されている．

唯一の欠点といえば，インタプリタの場合，ステップ数が大きいときLOAD時の翻訳時間がかかること．インクリメンタルコンパイル方式なので，翻訳ワークメモリ領域がテキスト領域の約3倍必要になることで，640KBでMS-DOSのシステム領域が大きいとデータ領域が十分にとれないことである．Pro版の場合は，この点を改善し，データ領域をEMSに移しているが逆にEMS必装になる．

BASIC/98のプログラミング環境

QuickBASICなどの統合環境とは，基本的にコンセプトが違うので，そのような環境は用意されていないが，従来のBASIC経験者は抵抗なく移行でき，また従来のものにない拡張編集機能，デバッグ機能が付加されている．このような機能は，BASIC/98起動時にパラメータを設定して希望する環境を構築できる．ここでは，BASIC/98 Fast V3.0版の仕様を中心に紹介したい．

起動時オプション

A>BASIC98 /M:・・ /B:・・ /F:・・ /E:・・ /C:・・ /W:・・ /T: RUN"・・"

このオプションは，次のように設定される．

/M:　ワーク領域，テキスト領域
/F:　同時オープンファイル数
/S:　レコード長サイズ
/C:　実行時／入力時コンパイル
/E:　拡張機能.EXE
/I:　データ入力時色，エラー設定
/W:　エラーメッセージの日本文，英文，コード表示
/T:　起動時実行ファイル指定
/B:　基本機能の組み込み
　±C　COM関連命令
　±M　MOUSE関連命令
　±G　GRAPHICS関連命令
　±E　拡張編集命令
　±D　拡張デバッグ命令
　±X　クロスリファレンス命令

ここでは，とくに基本機能の中で拡張編集，拡張デバッグ，クロスリファレンス関係について述べよう．

【FIND機能(＋E)】

これは小ステップでは不要であるが，数十ステップ以上になるとある文字列，変数名を画面スクロールで探すのは困難であるが，FINDで簡単に検索できる．

リスト1のように，二重引用符で囲まれた文字列を次々検索でき，その文字列が表示される．行番号で検索範囲を指定できる．

【REPLACE機能(＋E)】

プログラム中の任意の文字列を他の文字列で置き換える機能で，リスト2のような使い方が可能である．リスト1の120行目のPRINTをLPRINTに変換すると，リスト2のように実行結果を画面出力，プリンタ出力へ切り替えられる．

【HELP機能(＋E)】

統合環境では，この機能とエラーメッセージ機能が充実しているが，BASIC/98ではこのコマンドで必要なコマンドの説明を見ることができる．「マニュアル・イン・ソフト」形式である．たとえば，「FIND」についてHELPするとリスト3のようになる．

【変数値表示機能(＋D)】

デバッグ時にある文の実行時の変数値を知りたいことがある．PRINT!命令は，リスト4にあるように，変数値が2進数に対応する16進数表示とASCII表示で示される．

16進数は，右端が下位バイトなので，図1のように読み取る．

【追跡実行拡張機能(＋D)】

従来のBASICでも，TRON命令により実行経過を追跡できた．しかし，画面上が[<行番号>]の表示ですぐにいっぱいになり，実用的ではなかった．BASIC/98では

TRON,STEP　　一行実行

図1

```
A% :                0001      1
B! :            81280000      1.3125
C# :    8128000000000000      1.3125
D$ :            F1BC0003      F1BC番地から3バイト
```


BASICプログラミング用語集
ステップ　プログラムの命令語1個のこと．普通プログラムのサイズはステップで表現する．ただしBASICでは1行を1ステップと数える．

TRON,SLOW　　ゆっくり実行
TRON,BREAK　ブレークポイント実行

の3種類がある．実行例をリスト5に示す．

リスト5中185行の

REM $DEBUG PRINT A!

は，拡張デバッグ機能(＋D)時のみ実行できる命令で，185行でのAの値を表示できる．この機能を停止すると(−D)この文は単なるREM文となって実行に影響しない．

【クロスリファレンス機能(＋X)】

大きなサイズのプログラムになると変数リスト，参照行番号のリストが欲しくなる．このようなとき，リスト6ようにプログラムのLOAD後XREFコマンドを実行する．

このプログラム中で使われている変数のすべてについて，変数名，型，参照行番号が表示される．この例では分岐先がないが，GOTO文，GOSUB文などを用いると行番号，ラベル名を参照している行番号が表示される．

BASIC/98の構造化構文

識別名としての変数，関数，配列などの名前

構造化構文の紹介に入る前に，変数名などに用いられる識別名の文法について述べる．従来のBASICと異なり，識別名の文字列の長さに制限がないが，BASICでは最大文字長が255バイトなので，ANKモードで255文字までとなる．また，日本語文字も許されてい

リスト1

```
100 REM FIND
110 FOR I=1 TO 10
120   PRINT "FIND"
130 NEXT I
140 END
Ok
FIND "FIND"
100 REM FIND
Ok
FIND
120 PRINT "FIND"
Ok
FIND
Ok
```

リスト2

```
REPLACE "FIND","REPLACE"
100 REM REPLACE
120   PRINT "REPLACE"
Ok
REPLACE "PRINT","LPRINT"
120   LPRINT "REPLACE"
Ok
LIST
100 REM REPLACE
110 FOR I=1 TO 10
120   LPRINT "REPLACE"
130 NEXT I
140 END
Ok
```

リスト3

```
HELP ,"FIND"
*E1      FIND

FIND ["<検索文字列>"[,<行番号1>]
     [-[<行番号2>]],[A]]
<検索文字列>で指定した文字列を・・
・・・・・・・・略・・・・・・・・・
・・・・・・・・・・すべて表示する。

------------------------------------
Ok
```

リスト4

```
100 REM PRINT!
110 A%=1
120 B=A%+.3125
130 C#=B
140 D$="ABC"
150 PRINT ! A%,B,C#,D$
160 END
Ok
RUN
翻訳しました。
A% :  0100        1
B! :  00002881       1.3125
C# :  0000000000002881      1.3125
D$ :  0300BCF1      ABC
Ok
```

リスト5

```
100 REM DEBUG
110 X%=0
120 DO
130   A%=X% MOD 2
140   IF A%<>0 THEN
150     PRINT "奇数"
160   ELSE
170     PRINT "偶数"
180   END IF
185   REM $DEBUG PRINT ! A%
190   X%=X%+1
200 LOOP UNTIL X%=0
210 END
Ok
TRON,STEP
Ok
RUN
Compiled
[100]
[110]
[120]
[130]
[140]
[170]
偶数
[180]
[185]
A% :  0000      0
[190]
[200]
[130]
[140]
[150]
奇数
[160]
[180]
```

(以下略)

リスト6

```
LOAD "DEBUG"
LIST
100 REM DEBUG
110 X%=0
120 DO
130   A%=X% MOD 2
140   IF A%<>0 THEN
150     PRINT "奇数"
160   ELSE
170     PRINT "偶数"
180   END IF
185   REM $DEBUG PRINT ! A%
190   X%=X%+1
200 LOOP UNTIL X%=0
210 END
Ok

XREF
変数名対応リスト
A%          :  130,  140
X%          :  110,  130,  190,  200

分岐先対応リスト

Ok
```

るので２バイト文字で127文字までとなる．実際には，このように長い名前をつけることはないが，日本語文字変数によりプログラムの可読性が向上する．もちろん，開発時はANKで，完成時にREPLACEで変換すればよい．

リスト７にその例を示す．ところが，２バイト文字ならなんでもよいので，リスト８のように２バイトの空白や記号も変数名として許されてしまう．

次に構造化の問題であるが，従来のBASICの最大の問題点はGOTO文，IF文などの分岐先の問題である．これをまったく用いないで，プログラミングせよとまではいわないが，長く初心者の教育を行ってきた経験から，多少プログラミングができるようになると，これらのGOTO文などでプログラムの流れを勝手気ままにひっぱりまわし，結果としてザルソバいやスパゲッティになるという仕儀で，プログラミング教育の上から重大なことといえる．

判定反復構造（DOブロック構造）

最初の構造として判定反復を取り上げる．これには，処理の形態として処理の事前あるいは事後に判定する２通りがある．従来のBASICでは図２のような形式しかなく，IFの条件式に対して，WHILEの条件式は否定の形になる．前者は条件式成立で反復脱出，後者は条件式成立で反復実行であり，初心者にとってはまごつくところである．

これらの形式も可能であるが，BASIC/98では図３のDOブロック構造が望ましい．DO-LOOP文では４通りの組み合わせがあり，これですべての反復脱出が表現できる．

事前判定と事後判定の違いは，前者は<処理ブロック>の実行の前に判定されるため１回も<処理ブロック>が

リスト７

```
100 REM 変数識別名
110 円周率=3.14159
120 INPUT 半径
130 円周=2*円周率*半径
140 PRINT "円周＝";円周
150 END
Ok
RUN
? 10
円周＝ 62.8318
Ok
```

リスト８

```
REPLACE "半径","　　"
120 INPUT
130 円周=2*円周率*
Ok
LIST
100 REM 変数識別名
110 円周率=3.14159
120 INPUT
130 ＊＊=2*円周率*
140 PRINT "円周＝";＊＊
150 END
Ok
RUN
? 10
円周＝ 62.8318
Ok
```

図２

```
*LABEL1
 IF <条件式> THEN *LABEL2
    <処理ブロック>
 GOTO *LABEL1
*LABEL2
あるいは
 WHILE <条件式>
    <処理ブロック>
 WEND
```

図３

```
    ┌ WHILE ┐
DO  ┤       ├ <条件式>
    └ UNTIL ┘
   <処理ブロック>
LOOP
      [事前判定]

DO
   <処理ブロック>
      ┌ WHILE ┐
LOOP  ┤       ├ <条件式>
      └ UNTIL ┘
      [事後判定]
```

BASIC プログラミング用語集

ソースファイル　BASICではステートメントや関数を記述したテキストファイルのこと．インクルードファイルのようにオブジェクトにならないソースファイルもある．

実行されないこともあるが，後者は条件式の成否にかかわらず，少なくとも1回は処理ブロックが実行されることである．また，WHILEは成立時反復実行，UNTILは成立時反復脱出である．例文をリスト9に示す．

入力文を含む判定反復（無条件DOブロック）

従来のBASICでは，入力文を含む判定反復を構造化で表現するには多少無理があった．IF-GOTO文形式では，簡単に表現できるものがWHILE-WENDでリスト10のようになってしまう．

このような場合を無条件DOブロック構造で表わすと，リスト11のように表わせる．ここで，条件式成立で脱出するためにEXIT　DO文が用いられている．これは，IF-THEN文の条件式が成立するとDO-LOOP文を脱出し，LOOP文の次の行に実行が移る働きをもつ文である．どのようなアルゴリズムを用いてもよいが，無条件のDO-LOOP文はこのような入力文を含む反復に有効である．

この無条件のループを従来のBASICで実現すると，リスト12のような姑息なアルゴリズムになる．

リスト9

```
100 REM DO-LOOP
110 DEFINT I-N
120 I=1
130 DO WHILE I<10
140   PRINT I;
150   I=I+1;
160 LOOP
170 PRINT
180 END

Ok
RUN
Compiled
 1  2  3  4  5  6  7  8  9
Ok


100 REM DO-LOOP
110 DEFINT I-N
120 I=10
130 DO UNTIL I<1
140   PRINT I;
150   I=I-1;
160 LOOP
170 PRINT
180 END
Ok
RUN
Compiled
 10  9  8  7  6  5  4  3  2  1
Ok


100 REM DO-LOOP
110 DEFINT I-N
120 I=1
130 DO
140   PRINT I;
150   I=I+1;
160 LOOP WHILE I<10
170 PRINT
180 END

Ok
RUN
Compiled
 1  2  3  4  5  6  7  8  9
Ok


100 REM DO-LOOP
110 DEFINT I-N
120 I=10
130 DO
140   PRINT I;
150   I=I-1;
160 LOOP UNTIL I<1
170 PRINT
180 END
Ok
RUN
Compiled
 10  9  8  7  6  5  4  3  2  1
Ok
```

リスト10

```
100 REM IF-GOTO
110 *INPUT.LOOP
120  INPUT INDATA
130  IF INDATA=99 GOTO *LOOP.OUT
140    PRINT "INDATA=";
150    PRINT INDATA
160  GOTO *INPUT.LOOP
170 *LOOP.OUT

100 REM WHILE-WEND
110 INPUT INDATA
120 WHILE INDATA<>99
130   PRINT "INDATA=";
140   PRINT INDATA
150   INPUT INDATA
160 WEND
```

リスト11

```
100 REM DO-LOOP
110 DO
120   INPUT INDATA
130   IF INDATA=99 THEN EXIT DO
140     PRINT "INDATA=";
150     PRINT INDATA
160 LOOP
```

リスト12

```
100 REM DO-LOOP
110 FOR I%=0 TO 0 STEP 0
120   INPUT INDATA
130   IF INDATA=99 THEN LOOP.OUT
140     PRINT "INDATA=";
150     PRINT INDATA
160 NEXT I%
170 *LOOP.OUT
```

BASICプログラミング用語集
ソフトウェア割り込み　サービスコールとも呼ばれ，OSやBIOSで用意されている基本命令群を呼び出す方法

制御変数による繰り返し構造 (FOR ブロック構造)

繰り返し回数を指定して反復する方法がある．まずDO-REPEAT文について紹介しよう．この場合も図4に示す通り事前と事後の2通りの判定方法がある．

回数カウントの例文をリスト13に示す．回数のカウントの位置が事後の場合は，処理ブロックが1回実行されてから回数カウントになるので11回繰り返されている．

次に，従来のBASICでもよく使われるFORブロック構造であるが，BASIC/98ではとくにブロック途中の脱出方法にEXIT FOR文がある．

文の形式は図5の通りである．

一般には，一重ループでは問題が起こらないものの，多重ループの場合や複雑な判定の中に入るループの場合にトラブルが起こることがある．FOR-NEXT文は，普通FOR文の出現で繰り返し回数を計算し対応するNEXT文でカウントを行う．このとき，実行する順序から意図しないNEXT文と対応して，予期しない動作をすることがある．また，多重ループのネスト(入れ子)構造では，FOR文とNEXT文の出現により，スタック操作を行っているので，場合によってスタックが解放されず，スタックオーバーフローが発生することもある．

そこで途中脱出であっても，FOR-NEXTの対応をくずさないEXIT FOR文が用意されている．リスト14のような多重ループ脱出の例を示す．

ネストの最深部から構造的に脱出するには，すこし手間がかかる．一気にGOTO文で脱出しても，問題が起こらないように注意すればかまわない．

EXIT FOR文は，そのIF文の条件式が成立すると，そのブロックのNEXT文の次の命令文に実行が移る．同時にスタックは，一段解放されている．

選択構造 (IF ブロック構造)

従来のBASICによるプログラミングで，最悪のものがこの構造であった．たとえば，図6のような場合は，何ら問題は起こらないし，シンプルな場合はあえて構造化しないほうがよい．なぜなら，図7のようにくどくなるからである．しかし，一般的には選択構造のアルゴリズムには，図8のような構文を基本的に用いるべきである．

このような構文において，＜処理ブロック＞にはどのような長いステップの文が挿入されてもよい．

図4

```
DO REPEAT 〈回数〉
  〈処理ブロック〉
LOOP
                    〈事前判定〉

DO
  〈処理ブロック〉
LOOP REPEAT 〈回数〉
                    〈事後判定〉
```

図5

```
FOR 〈制御変数〉=〈始値〉 TO 〈終値〉 STEP 〈増分〉
      〈処理ブロック〉
    IF 〈条件式〉 THEN EXIT FOR
      〈処理ブロック〉
NEXT 〈制御変数〉
```

図6

```
IF 〈条件式〉 THEN 〈代入文〉
```

図7

```
IF 〈条件式〉 THEN
   〈代入文〉
END IF
```

図8

```
IF 〈条件式〉 THEN
   〈処理ブロック〉
ELSE
   〈処理ブロック〉
END IF
```


BASIC プログラミング用語集
ダイナミック配列　プログラムの実行中にメモリが割り当てられる配列のこと．割り当てられるまでアドレスを知ることはできない．

リスト13

```
100 REM DO-REPEAT
110 DEFINT I
120 I=10
130 DO REPEAT 10
140   PRINT I;
150   I=I-1
160 LOOP
170 PRINT : PRINT I
180 END
Ok
RUN
Compiled
 10  9  8  7  6  5  4  3  2  1
 0
Ok
100 REM DO-REPEAT
110 DEFINT I
120 I=10
130 DO
140   PRINT I;
150   I=I-1
160 LOOP REPEAT 10
170 PRINT : PRINT I
180 END
Ok
RUN
Compiled
 10  9  8  7  6  5  4  3  2  1  0
-1
Ok
```

リスト14

```
100 REM FOR-NEXT-EXIT
110 FOR I=0 TO 1
120   FOR J=0 TO 1
130     FOR K=0 TO 1
140       M=4*K+2*J+I
150       IF M=5 THEN EXIT FOR
160     NEXT K
170     IF M=5 THEN EXIT FOR
180   NEXT J
190   IF M=5 THEN EXIT FOR
200 NEXT I
210 PRINT I;J;K
220 END
Ok
RUN
Compiled
 1  0  1
Ok
```

リスト15

```
100 REM IF-THEN-ELSE
110 FOR I=1 TO 5:X(I)=2*I:Y(I)=2*I+1:NEXT I
120 INPUT A,B,C,D
130 IF A>0 THEN FOR I=1 TO 9 :IF D>0 THEN
 140 ELSE NEXT I :IF B>0 THEN IF C>0 THEN
 FOR J=1 TO 5:PRINT X(J);:NEXT J ELSE FOR
 K=1 TO 5:PRINT Y(K);:NEXT K:PRINT
```

リスト16

```
100 REM IF-THEN-ELSE-END IF
110 FOR I=1 TO 5
120   X(I)=2*I
130   Y(I)=2*I+1
140 NEXT I
150 INPUT A,B,C,D
160 IF A>0 THEN
170    FOR I=1 TO 5
180      IF D>0 THEN EXIT FOR
190    NEXT I
200    IF NOT D>0 THEN
210      IF B>0 THEN
220         IF C>0 THEN
230           FOR J=1 TO 5
240             PRINT X(J);
250           NEXT J
260         ELSE
270           FOR K=1 TO 5
280             PRINT Y(K);
290           NEXT K
300         END IF
310      END IF
320    END IF
330 END IF
340 PRINT "END"
350 END
```

従来のBASICでは，IF-THEN-ELSE文を1行で表現しなければならず，THENブロックやELSEブロックに，さらにIFブロックを含むような複雑な構造の場合，極めて読みにくいものになる．その例文をリスト15に示す．プログラム自体は，あまり意味のあるものではないが構造化という点に注目してほしい．

この130行を1行で表わしているものを，構造的に表現するとリスト16のようになる．プログラムは長くなるが，プログラムの可読性はいずれが優れているか一目瞭然である．

多選択構造（ELSEIFブロック）

次に選択条件が3つ以上にわたり，それぞれにおいていろいろな処理がある場合，IF-THEN-ELSE-END IFの構文では複雑になるので，ELSEIF構文が登場した．その構文は図9の通りである．

たとえば，リスト17のようなランキング処理のアルゴリズムに適している．

多選択構造（CASEブロック構造）

多選択構造には，判定式が式の形をとらないで値をとる選択がある（リスト18）．

従来のBASICでは，これらの選択構造をON GOTO，ON GOSUB形

図9

```
IF <条件式> THEN
   <処理ブロック>
ELSEIF
   <処理ブロック>
ELSEIF
   <処理ブロック>
・・・・・
・・・・・
ELSE
   <処理ブロック>
END IF
```

リスト18

```
100 REM CASE
110 INPUT SELECT.MENU$
120 SELECT SELECT.MENU$
130   CASE "1" : GOSUB *A
140   CASE "2" : GOSUB *B
150   CASE "3" : GOSUB *C
160   CASE "4" : GOSUB *D
170   CASE ELSE
180     PRINT "INPUT ERROR"
190 END SELECT
200 END
```

リスト17

```
100 REM ELSEIF
110 INPUT TEN
120 IF TEN>=80 THEN
130   HYOKA$="A"
140 ELSEIF TEN>=70 THEN
150   HYOKA$="B"
160 ELSEIF TEN>=60 THEN
170   HYOKA$="C"
180 ELSE
190   HYOKA$="D"
200 END IF
210 PRINT HYOKA$
220 END
```

リスト19

```
100 REM CASEIS CONDITION
110 INPUT TEN
120 SELECT
130   CASEIS TEN>=80
140     HYOKA$="A"
150   CASEIS TEN>=70
160     HYOKA$="B"
170   CASEIS TEN>=60
180     HYOKA$="C"
190   CASE ELSE
200     HYOKA$="D"
210 END SELECT
220 END
```

式により，すべて行番号，ラベル指定で実現してきた．また，値による選択のみならず条件式によるCASE選択も可能である（リスト19）．

関数定義構造（DEFFNブロック構造）

BASIC/98の構造化は関数定義にもおよび，従来の利用者定義関数DEFFN文では，定義関数は式を並立させるのみで，条件判定を含むような形式は許されなかった．ここでは，DEFFN-END DEF文の構文例を紹介する（リスト20）．従来は，このようなアルゴリズムは，GOSUB文によるサブルーチン構造で表現せざるを得なかった．

また，条件によってDEFFNブロックを脱出することも可能である（図10）．

この関数名のあとにつく引数としての変数名は，この関数内で有効なので，ある意味で独立である．しかし，引数で定義しない関数内の変数は，主プログラムと共通変数となりDEFFNブロックは完全独立ではない．

関数定義構造（FUNCTIONブロック構造）

BASIC/98では，さらに定義関数のブロックを完全に独立化するために，図11のFUNCTIONブロック構造が用意されている．DEFFNブロック構造と異なるのは，主プログラムのあとにFUNCTIONブロックを置く点である．

FUNCTIONブロック内で定義されている引数変数のXと，内部変数Yは主プログラムとは無関係で，このブロックは主プログラムから完全に独立である（リスト21）．よって，いろいろな関数を定義して，関数ファイルとして保存しておくことができる．その意味で定義関数に汎用性が生じ，ライブラリ化していろいろな主プログラムにMERGEできる．ただし，関数はその性質から値を1つしか返せない．

サブルーチン構造（SUBブロック構造）

従来のBASICでは，GOSUB文によるサブルーチン構造であったため，単に呼び出した位置の次に戻る命令をもったGOTO文に過ぎなかった．このため，サブルーチン部分に用いられている変数名は，すべて主プログラム領域の変数名と共用であり，まったく独立になっていなかった．そのため，主プログラムごとに同じ働きのサブルーチンを用意しなければならなかった．

BASIC/98では，機械語サブルーチンを呼び出すCALL文以外に，図12

BASICプログラミング用語集
デバイス　CPUに接続されるディスプレイ・キーボード・FDD・メモリなどすべての周辺機器をいう．

のような独立なSUBブロック構造をもつサブルーチンが用意された．

主プログラムのCALL文の()内の引数(実引数)の変数X，YとSUB文の()内の引数（仮引数）P，Qとが順序，型，引数の個数を対応させて結合している(リスト22)．主プログラムとサブルーチン間のデータのやりとりは，すべてここで行われるので相互に完全に独立である．

以上で，従来のBASICに加えてBASIC/98で採用された構造化構文を，例文を交えて紹介した．QuickBASICと比べても，この点で決して遜色はないが，QuickBASICにあってBASIC/98にないものは，レコード型定義TYPE-END TYPE位であり，その意味でQuickBASICはC言語に近い仕様で，より開発向きとなっている．しかし，統合環境の定義，習熟の手間，コンパンル形式，専用エディタの習熟などを考えると，初心者教育用としては必ずしも勧められない．

追加された関数など

BASIC/98で追加されたいくつか

図10

```
DEF FN<関数名>(<引数群>)
    [関数定義ブロック]
  IF <条件式> THEN EXIT DEF
    [関数定義ブロック]
END DEF
```

図11

```
FUNCTION FN<関数名>(<引数群>)
    [関数定義ブロック]
  IF <条件式> THEN EXIT FUNCTION
    [関数定義ブロック]
END FUNCTION
```

図12

```
REM 主プログラム
  ....
CALL *<サブルーチン名>(<引数群>)
  ....
END

SUB *<サブルーチン名>(<引数群>)
    [サブルーチンブロック]
  IF <条件式> THEN EXIT SUB
    [サブルーチンブロック]
END SUB
```

リスト20

```
100 REM DEFFN-END DEF
110 DEF FNF(X)
120    IF X>=0 THEN
130       FNF=3*X+4
140    ELSE
150       FNF=-2*X-4
160    END IF
170 END DEF
200 REM MAIN
210 INPUT Z
220 Y=FNF(Z)
230 PRINT Z,Y
240 END
```

リスト21

```
100 REM MAIN
110 INPUT X
120 Y=FNF(X)
130 PRINT X,Y
140 END

200 REM FUNCTION END FUNCTION
210 FUNCTION FNF(X)
220    IF X>=0 THEN
230       Y=3*X+4
240    ELSE
250       Y=-2*X-4
260    END IF
270    FNF=Y
280 END FUNCTION
```

リスト22

```
100 REM MAIN
110 INPUT X
120 CALL *ABSF(X,Y)
130 PRINT X,Y
140 END

200 REM SUBROUTINE
210 SUB *ABSF(P,Q)
220    IF P>=0 THEN
230       Q=3*P+4
240    ELSE
250       Q=-2*P-4
260    END IF
270 END SUB
```


デバッガ　トレーサとも呼ばれ，プログラムのバグを発見するソフトウェアまたはハードウェア．ソースレベルでのデバッグを可能にしているものも少なくない．

の関数を例文を中心に紹介する(リスト23〜26).

CAPS$関数は英小文字を英大文字にする関数である. NPIECE関数は文字列中の検索文字の個数, PIECE$関数は指定文字にはさまれた文字列の切り出し, UBOUND/LBOUND関数は配列の上下限を示す.

ファイル管理の追加機能

ライブラリ関数, サブルーチンライブラリのMERGE

独立なFUNCTIONライブラリによる関数プログラムや, サブルーチンプログラムをファイルとして用意されているとき, これを自分で作った主プログラムに結合したいなら, これらのライブラリやサブルーチンプログラムは, ASCIIタイプでSAVEしておく必要がある.

プログラムAにサブルーチンB, ファンクションCを結合する例文をリスト27に示す. オプションの「,A」が, 行番号自動割付タイプのMERGEである. MERGEされた結果は, リスト28のように, 自動的に行番号が調整されるので, ファイルとして保存しておくことができる. このプログラムのクロスリファレンスをXREFすると, リスト29のようになる.

エディタ作成ソースファイルの格納

人によっては, 使い慣れたエディタでソースプログラムを作成したいという欲求がある. このような場合も, リスト30のようにすれば可能である.

リスト23 CAPS$関数(大文字変換)

```
100 REM CAPS$
110 INPUT C$
120 D$=CAPS$(C$)
130 PRINT D$
140 END
Ok
RUN
Compiled
? abc
ABC
Ok
```

リスト24 NPIECE関数(文字個数検索)

```
100 REM NPIECE
110 X=NPIECE("ABCDEABC","A")
120 PRINT X
130 END
Ok
RUN
Compiled
 2            Aが2個あった
Ok
```

リスト25 PIECE$関数(文字列切取り)

```
100 REM PIECE$
110 A$=PIECE$("PQ:ABC:XY",":",3)
120 PRINT A$
130 END
Ok
RUN
Compiled
ABC         :で囲まれた文字を3文字
Ok          分切取る。
```

リスト26 UBOUND関数(配列上限)/LBOUND関数(配列下限)

```
100 REM MAIN
110 OPTION BASE 1
120 DIM B(20)
130 CALL *HAIRETSU(B())
140 END

200 SUB *HAIRETSU(A())
210 L%=LBOUND(A,1)
220 U%=UBOUND(A,1)
230 DIM A(U%)
230 PRINT L%,U%
240 END SUB
Ok
RUN
Compiled
 1         20
Ok
```

LOAD時の「,L」オプションがエディタ作成ソースの読み込みを可能にする.

ランダムファイルの可変レコード長

従来のBASICでは, ランダムファイルの通常のフィールド長は256バイト固定であった. このため, レコード長が256バイトより小さければ, 余った部分がムダになり, 大きいときは数フィールドを1レコード長とするアルゴリズムを考えねばならなかった. また, 256バイトを分割してレコードを割り当てるにも, 同様にそのアルゴリズムを必要とする. その例をリスト31, 32に示す. 1レコードが32バイトの場合と512バイトの場合である.

これらが起動オプションの設定だけで, リスト33, 34のように簡便になる. 512バイトについては, 文字列の最大長が255バイトなので, 最後の1バイトは改行コードにしてある.

BASIC98Cコンパイラ環境

BASIC/98のコンパイラ環境に少なくとも必要なファイルは

BASIC98C.EXE
BASIC98L.EXE
BASIC98L.LIB
B98GRAPH.EXE
LINK.EXE
BATGEN.EXE

BASICプログラミング用語集
ドラッグ　マウスのボタンを押しながらマウスを動かすこと. ウィンドウシステムなどで対象の移動や, 範囲指定に使われる.

リスト 27

```
LOAD "A"
LIST
100 REM A(MAIN)
110 INPUT N,R,P
120 CALL *KAI(N,KN)
130 CALL *KAI(R,KR)
140 CALL *KAI(N-R,KNR)
150 NCR=KN/KR/KNR
160 B=FNB(NCR,P,N,R)
170 PRINT N,R,NCR,P,(1-P),B
180 END
Ok
LOAD "B"
LIST
100 REM B(SUB)
110 SUB *KAI(M,K)
120 K=1
130 FOR I=1 TO M
140   K=K*I
150 NEXT I
160 END SUB
Ok
LOAD "C"
LIST
100 REM C(FUNC)
110 FUNCTION FNB(M,P,N,R)
120   B=M*P^R*(1-P)^(N-R)
130   FNB=B
140 END FUNCTION
Ok
LOAD "A"
MERGE "B",A
MERGE "C",A
Ok
```

リスト 28

```
LIST
100 REM A(MAIN)
110 INPUT N,R,P
120 CALL *KAI(N,KN)
130 CALL *KAI(R,KR)
140 CALL *KAI(N-R,KNR)
150 NCR=N/R/(N-R)
160 B=FNB(NCR,P)
170 PRINT N,R,B
180 END
190 REM B(SUB)
200 SUB *KAI(M,K)
210 K=1
220 FOR I=1 TO M
230   K=K*I
240 NEXT I
250 END SUB
260 REM C(FUNC)
270 FUNCTION FNB(M,P)
280   B=M*P^N*(1-P)^(N-R)
290   FNB=B
300 END FUNCTION
```

リスト 29

```
変数名対応リスト
B!        : 160, 170, 280, 290
B!()      : 160, 270
I!        : 220, 230, 240
K!        : 200, 210, 230
KN!       : 120, 150
KNR!      : 140, 150
KR!       : 130, 150
M!        : 200, 220, 270, 280
N!        : 110, 120, 140, 160
            170, 270, 280
NCR!      : 150, 160, 170
P!        : 110, 160, 170, 270
            280
R!        : 110, 130, 140, 160
            170, 270, 280

分岐先対応リスト
200,*KAI  : 120, 130, 140
```

リスト 30

EDITORで以下のように入力する

```
REM MAIN
DEFINT I-N
INPUT MAX,KOSU,INIT
RANDOMIZE INIT
FOR I=1 TO KOSU
  ME=INT(RND*MAX+1)
  PRINT ME;
  LF=I MOD 20
  IF LF=0 THEN PRINT
NEXT I
END
```

保存ファイル名例<RANSU.BAS>

```
LOAD "RANSU",L
Ok
 10 REM MAIN
 20 DEFINT I-N
 30 INPUT MAX,KOSU,INIT
 40 RANDOMIZE INIT
 50 FOR I=1 TO KOSU
 60   ME=INT(RND*MAX+1)
 70   PRINT ME;
 80   LF=I MOD 20
 90   IF LF=0 THEN PRINT
 100 NEXT I
 110 END
RENUM 100
Ok
```

リスト 31

```
100 REM SHORT-RECORD 32BYTES 分割
110 DEFINT I-N
120 DEFSTR A-D
130 OPEN "SDATA.DAT" AS #1
140 ISR=0
150 FOR I=1 TO 8
160   FIELD #1,ISR AS DMY,32 AS D(I)
170   ISR=ISR+32
180 NEXT I
190 DO
200   INPUT "RECORD NO=";IR
210   IF IR=0 THEN EXIT DO
220     MR=IR MOD 8
230     IF MR=0 THEN MR=8
240     IF MR=8 THEN K=0 ELSE K=1
250     NR=IR¥8+K
260     LSET D(MR)="12345678901234567890123456789012"
270     PUT #1,NR
280     GET #1,NR
290     DATAIR=D(MR)
300     PRINT NR,MR,D(MR)
310 LOOP
320 CLOSE
330 END
```

リスト 32

```
100 REM LONG-RECORD 512バイト連結
110 DEFINT I-N
120 DEFSTR A-D
130 OPEN "LDATA.DAT" AS #1
140 CR=CHR$(13)
150 FIELD #1,255 AS D,1 AS CR
160 DO
170   INPUT "RECORD NO=";IR
180   IF IR=0 THEN EXIT DO
190     NR=2*IR-1
200     FOR OFFSET=1 TO 512¥256
210       OFFR=OFFSET-1
220       MR=NR+OFFR
230       LSET D=STRING$(255,"X")
240       PUT #1,MR
250       GET #1,MR
260       DATAIR=D
270       PRINT MR,DATAIR
280     NEXT OFFSET
290 LOOP
300 CLOSE
310 END
```

リスト33

BASIC98 /S:32 で起動

```
100 REM SHORT-RECORD /S:32
110 DEFINT I-N
120 DEFSTR A-D
130 OPEN "SDATA2.DAT" AS #1
140 FIELD #1,32 AS D
150 DO
160   INPUT "RECORD NO=";IR
170   IF IR=0 THEN EXIT DO
180     LSET D="12345678901234567890123456789012"
190     PUT #1,IR
200     GET #1,IR
210     DATAIR=D
220     PRINT IR,D
230 LOOP
240 CLOSE
250 END
```

リスト34

BASIC98 /S:512で起動

```
100 REM LONG-RECORD /S:512
110 DEFINT I-N
120 DEFSTR A-D
130 OPEN "LDATA2.DAT" AS #1
140 CR=CHR$(13)
150 FIELD #1,255 AS D1,1 AS CR,255 AS D2,1 AS CR
160 DO
170   INPUT "RECORD NO=";IR
180   IF IR=0 THEN EXIT DO
190       LSET D1=STRING$(255,"X")
200       LSET D2=STRING$(255,"X")
210       PUT #1,IR
220       GET #1,IR
230       DATAIR1=D1
240       DATAIR2=D2
250       PRINT IR,DATAIR1,DATAIR2
260 LOOP
270 CLOSE
280 END
```

から構成されており，リスト35のようなバッチファイルによりコンパイルできる．

[/L]のオプションを設定すると，BASIC98C.LIBの組み込みをしない．ライブラリの組み込みを行うと，生成されたEXEファイルは単独で実行可能であるが，組み込まないときは，ライブラリの実行ドライバBASIC98L.EXEが同じディレクトリ内に必要になる．両者それぞれのコンパイル結果のファイルをリスト30（RANSU.BAS)の例で比べよう（FM-R版，図13)．

ファイルサイズからは，明らかにライブラリを組み込まないほうがよいが，EXEファイルだけで動かすときは，組み込みのほうが取り扱いが簡単である．

BASIC/98のメモリ管理

BASIC/98の特徴は，メモリの管理を明確にしていることである．そのレイアウトは図14の通りである．

チャイルドプロセス，機械語プログラム，利用者スタックの各領域は，CLEAR文によってデフォルト値を変更可能である．また，利用者プログラム領域とワーク領域は，起動時のオプションで変更可能である．この値とMS-DOSが占有する領域から，利用者が利用できる配列データ領域，シンボルテーブル，ストリングデータ領域が決定される．そのため，場合によっては，MS-DOSのV.3.1あるいはV.2.11がV.3.3より望ましいこともある．

入出力バッファ領域も，起動時のファイル数の設定で変動する．利用者プログラム領域は，関係する領域を含めて57KBであるので，BASICのプログラムの大きさはおよそ50KB程度までとなる．これは，ワーク領域の関係からもこの程度が限界である．これ以上のサイズになるときは，プログラムをモジュール化しCHAIN命令などを用いて，プログラムの分割をしなければならない．

リスト35

```
BASIC98C %1.BAS %1.OBJ [/L]
IF ERRORLEVEL 1 GOTO ERRS
LINK %1.OBJ,%1.EXE,%1.MAP,BASIC98C
IF ERRORLEVEL 1 GOTO ERRS
%1.EXE
:ERRS
```

BASIC/98Pro環境

実用的なプログラムの開発用にBASIC/98Proが提供されている．これは，EMSが必要になるが，先のメモリレイアウトのうち，配列データ領域とワーク領域の中間コード領域が

BASIC プログラミング用語集
トレース　プログラムの実行に合わせて，CPUや周辺機器の状態を表示・記録すること．BASICでは各命令を実行する度に各変数の変化を調べ表示できる処理系もある．

EMM配下におかれる．そのため，配列データは以下の通りに確保できる．

整数型	128KB
文字型	256KB
単精度型	256KB
倍精度型	512KB
総計	1152KB

それぞれ要素数の和で65536個が上限になる．

もうひとつの特徴は，拡張エディタである．インタプリタを起動したのち，EDITコマンドを与えると拡張エディタ画面に入る．これは，BASIC/98専用のエディタで，ファンクションキーとプルダウンメニューでプログラムの編集を行うことが可能となっている．2つのウィンドウを操作可能で，行番号を自動生成するのでエディタ感覚でプログラム開発ができる．エディタを終了脱出すると，BASICのコマンド入力待ち状態になっているので直ちに実行可能である．

まとめ

わが国唯一の国産BASICであるBASIC/98について，従来のBASICととくに異なる点を中心に解説してきた．開発専用パッケージまで用意してあるものの，本格的な開発をするのならばC言語のほうがいいような気がする．BASIC/98の利点は，筆者の学校のように複数のメーカーのパソコンを使用している環境で，共通にN88-BASIC+構造化BASICが実現できる点であり，やはり「教育用BASICの本命」というのが結論である．もし，今後教育現場でBASICを教えられる，あるいは今教えておられるなら，直ちにBASIC/98を採用されることをお奨めする．

● BASIC/98商品の構成

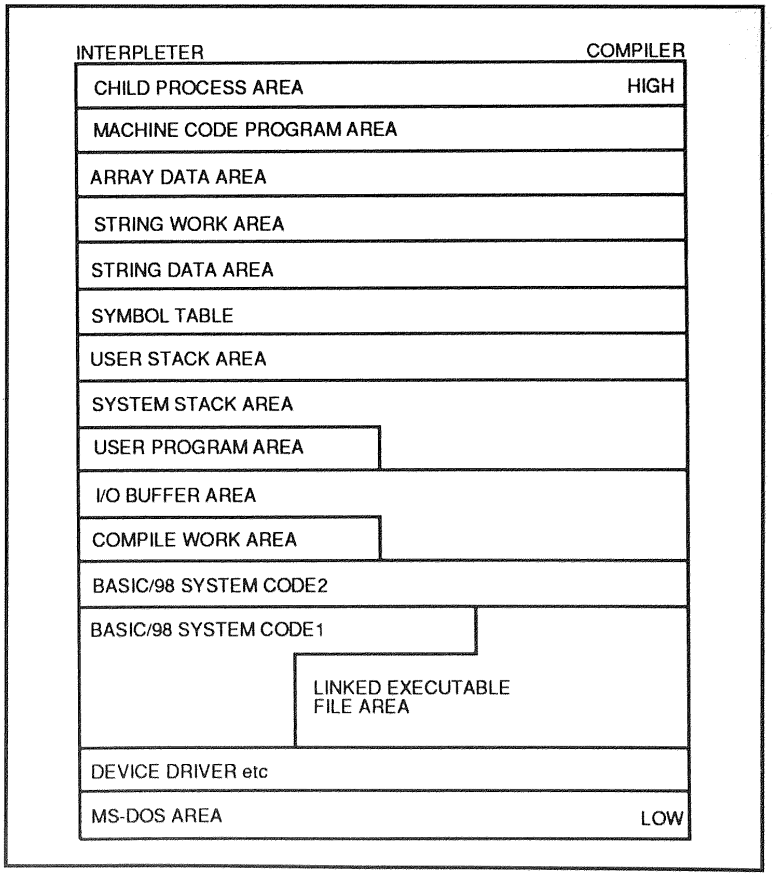
BASIC/98 Fast インタプリンタ	17,000円
BASIC/98 Fast コンパイラ	27,000円
BASIC/98 Pro インタプリンタ	47,000円
BASIC/98 Pro コンパイラ	57,000円

図13

```
  RANSU   BAS      206 Bytes
組み込みの場合（／Lなし）
  RANSU   OBJ     3838 Bytes
  RANSU   MAP     2420 Bytes
  RANSU   EXE   125922 Bytes
組み込みなしの場合（／Lあり）
  RANSU   OBJ     1380 Bytes
  RANSU   MAP      384 Bytes
  RANSU   EXE     4081 Bytes
```

FM-R版BASIC/98コンパイラ

図14



ハードウェア割り込み　ハードウェアからの要求で起こる割り込み．この割り込みが発生するとCPUは現在の処理を中断し，割り込みサービスルーチンへ制御を移す．

特集……やっぱり僕らはBASIC!

生みの親が作った由緒正しい処理系

True BASIC

編集部

Beginner's
All-purpose
Symbolic
Instruction
Code

1960年代半ばにダートマス大学のJ.G.Kemeny とT.E.Kurtz によって，学生の教育用に考案された言語BASIC (Beginner's All-purpose Symbolic Instruction Code) は，その名前のとおりコンピュータの基礎を学ぶものが最初に出会う基本的な言語として世界中に広まることとなった．その後，数々の問題点が指摘され，教育用言語としての地位はPASCALやLOGOに，開発用言語としての地位はCに，とって代わられた感さえある．このような状況のもと，BASICの生みの親である両氏自らが，既存のBASIC処理系の問題点を検討した上で作り上げた新しいBASICがTrueBASICだ．このTrueBASICがこのたび日本語化され，NECのPC-9800シリーズ用「日本語TrueBASIC V3.0J」として（株）クレオから発売された．TrueBASICとはどんなBASICなのか，その文法的な特徴と，日本語TrueBASICの操作方法の概要を紹介する．

TrueBASICとは

BASICの生みの親であるJ.G.Kemenyと T.E.Kurtzは，その著書「BACK TO BASIC」の中で，True BASIC誕生の経緯を語っている．これによれば，BASICという言語を考案するにあたっては，8つの原則から出発したのだという．8つの原則とは次のようなものである．

1．初心者が楽に覚えられること
2．汎用言語であって，どんなプログラムでも書けること
3．高度な機能は追加機能であること，したがって，もしお金がかかるのなら初心者ではなく熟練者が負担すること
4．利用者とコンピュータの対話機能を十分に生かしたものとすること
5．わかりやすく，親切なエラーメッセージが出ること
6．小さなプログラムなら迅速に応答できること
7．ハードウェアの知識は不要であること
8．利用者がオペレーティングシステムに立ち入らなくてよいこと

ところが，マイクロコンピュータの登場した時期に，限られた能力しかないマシンにBASICを搭載するため行われた数々の妥協が，そのままBASICという言語の特徴として今日にいたるまで保持されてしまった．そこにおいて，BASIC言語が誕生した当初の哲学はすっかり見失われてしまった．このような状況を憂えた両氏が，マイクロコンピュータ用にもう一度BASICを構築し，これに「True」(本物の) BASICと名付けたのである．

TrueBASICは，その誕生の経緯を見てもわかるように，従来のBASICの問題点を分析した上で作成されている．したがって，特徴として次のような点をあげることができる．

①構造化プログラミング言語である

TrueBASICは，構造化プログラミングの考え方を取り入れている．TrueBASICには，さまざまな機能をもった関数が組み込まれているが，これとは別にユーザーが独自に関数を定義することができる．大きなプログラムでは，関数やサブルーチンをいくつかのプログラムユニットに分割してライブラリとして利用することができる．また，モジュールという考え方が導入されており，モジュール内の変数をPUBLICに定義して外部から参照できるようにしたり，モジュール内の関数やサブルーチンをPRIVATEに定義して，外部から隠ぺいしたりするといったことも可能である．

従来BASICの欠点として指摘されてきた「GOTO文」も，TrueBASICではSELECT　CASE構文やDO WHILEループ構文，DO UNTIL構文などの新しい構文を導入したこともあって，まず使う必要がない．そもそも行番号すら必要ないのである．

②グラフィックスが利用できる

グラフィックス表示のできるディスプレイは，ハードウェアの発達によってBASIC言語が誕生した当初とは比較にならないほど身近なものとなった．そこで，TrueBASICには，パソコンの機能を最大限に利用できる2次元グラフィックス機能が用意されている．

③機種に依存しない汎用性をもつ

TrueBASICで書かれたソースプログラムは，機種によって記述を変更する必要はない．米国では，すでにMacintosh用，IBM PC用，UNIX用など各種コンピュータ用が発売されている．

さらに，「日本語TrueBASIC V3.0 J」(以下日本語TrueBASICと略)では，次のような特徴をあげることができる．

④エディタとコンパイラが一体化した快適な操作性

J.G.KemenyとT.E.Kurtzはリファレンスマニュアルの中でTrueBASICの特徴として，「プログラムの書き込みと修正が簡単なエディタがついている」ことをあげている．日本語TrueBASICには，マルチファイルの編集ができるエディタが付属しており，プルダウンメニューやファンクションキーを使って，プログラムの編集だけでなくコンパイルやインタプリタの実行なども行えるようになっている．

⑤ツールキットによる拡張性

日本語TrueBASICの機能を拡張する各種のツールキットを購入すれば，複雑なグラフィックスや科学技術演算などの関数やサブルーチンを利用することができる．

TrueBASICの文法上の特徴

まず，TrueBASICの特徴を文法の面から見てみる．文法上の特徴を取り

上げなければならないということ自体，今日BASICという言語のおかれた状況を表しているというべきだろう．本来プログラミング言語とは，言語さえ決まれば文法はひとつに決まるものであって，それぞれの処理系による違いはマシンに依存する機能など枝葉の部分になってくるはずであるが，BASICの場合はパソコンの普及とともに多数の方言が生まれてしまい，同じBASICという名前でも，実態はどれも別々である．TrueBASICでは，原点に立ち戻って，言語仕様はハードウェアに依存しないものであるとうたっている．

TrueBASICはひとことでいうと，たいへんシンプルな作りになっている．シンプルであることと高機能であることは，一見相反することのように思われるが決してそうではない．本稿冒頭にあげた8つの原則の3番目「高度な機能は追加機能であること，したがって，もしお金がかかるのなら初心者ではなく熟練者が負担すること」はそのことを示している．BASICのPRINT文とC言語のprintf文を比較してみればわかるが，PRINT文では後ろに変数を書きさえすれば，整数は整数として，小数部は6桁まで表示して改行される（もちろん必要とあらば，書式つきのPRINT USING文を書くこともできる）．printf文は，数値出力のさまざまな形態が指定できるが，その指定は省略できない．デフォルトである程度の機能が実現できる，必要ならオプションでさらに詳細な指定も行えるというのが，BASICの設計思想である．この思想は，TrueBASICでもいたるところで発揮されている．

また，本来ダートマス大学版のBASICに備わっていたにもかかわらず，パソコンBASICでは削り取られてしまった行列演算の機能なども，TrueBASICでは復活している．

さらに，BASIC誕生以来すでに30年近い年月を経たことから，TrueBASICでは現代的な改良がなされている．構造化プログラミングの採用，強力なグラフィックス機能の導入などである．

書式の特徴

TrueBASICで書かれたプログラムと，従来型のBASICで書かれたプログラムのサンプルをリスト1に示した．顕著な特徴はまず「行番号がない」ことである．そして，「大文字・小文字を混ぜて使うことができる」．さらに，自由に空白行を挿入することができるという特徴もある．これらの特徴はすべてプログラムの書きやすさ，読みやすさに通じるものである．

また，TrueBASICでは原則としてLETの省略はできない．「各文は必ずキーワードからはじまる」というのがTrueBASICの大原則である．ただし，既存の処理系の中にはLET文を省略できるものも多く，これらの処理系で作ったBASICプログラムを実行したいときの便宜をはかるためからか，プログラムの先頭「OPTION NOLET」と記述すれば，LET文を省略してもよいことになっている．

データ型

TrueBASICのデータ型は，数値と文字列の2種類しかない．マイクロソフト社のQuickBASICが，パソコンのハードウェアアーキテクチャに密着した形で整数，長整数，単精度浮動小数点数，倍精度浮動小数点数と4つの数値データ型を扱うのとは大きな違いである．J.G.KemenyとT.E.Kurtzは，プログラミング言語はハードウェアとは独立すべきであり，利用者がコンピュータ内部のハードウェアについて知る必要はないという考えをとっていた．本稿の冒頭であげた原則の7，8がこれにあたる．つまり，相手がUNIXマシンであれ，パソコンであれ，プログラマはその変数が数値であるということさえわかっていればプログラムは動くし，動かなければならないということである．

構文

条件判断構文として，IF構文とSELECT CASE構文が使用できる．それぞれの文の終わりは，必ずEND IF文，END CASE文を記述することとし，書きやすさ，読みやすさが高め

リスト1

```
! サイン関数
!
def F(x) = Sin(x) + Cos(2*x)

let k = 3*pi

set window -k, k, -2, 2
set color "red"

for x = -k to k step .1
    plot x, F(x);
next x
end
```

リスト1'

```
100 !  サイン関数
110 !
120 DEF FNF(X) = SIN(X) + COS(2*X)
130 !
140 DEF FNSCALEX(X) = X/640 * 2 * K - K
150 !
160 DEF FNSCALEY(Y) = 400 - (Y+2)/4 * 400
170 !
180 LET K = 10
190 !
200 SCREEN 3
210 !
220 FOE X = 0 TO 640 STEP 5
230     Y = FNSCALEY(FNF(FNSCALEX(X))
240     IF X>0 THEN LINE(OX,OY)-(X,Y),2
250     OX = X: OY = Y
260 NEXT X
270 END
```

BASIC プログラミング用語集
配列　データの並べ方．配列を利用することによりデータを的確に指定できるようになる．テキストやグラフィックス画面も通常は二次元配列で表現される．

られている．

ループ構文として，FOR NEXT構文とDO LOOP構文が使用できる．それぞれのループでは，EXIT FOR文，EXIT DO文を使ってループを抜けることができるなど，行番号を使わなくても文の制御を移せるように配慮されている．

図1　DO UNTIL ループと DO WHILE ループ

```
DO WHILE<条件1>          条件1が真のとき処理1をループ
    〜
  <処理1>                条件1が偽になったらループを抜ける
    〜
LOOP
```

```
DO UNTIL<条件2>          条件2が真になったらループを抜ける
    〜
  <処理2>                条件2が偽の間は処理2をループ
    〜
LOOP
```

図2　無限ループからの脱出

[無限ループ]

```
DO
   PRINT"続けます"
LOOP
```

[ループからの脱出]

```
DO
   PRINT"続けますか？Y/N"
   INPUT  ans$
   IF ans$ = "N" THEN EXIT DO
LOOP
```

行列演算のサポート

TrueBASICには，行列演算の機能が用意されている．ここでは例として，次のような3元1次連立方程式の解を求めるプログラムを作成してみる．

$$x+2y+4z=17$$
$$2x+3y+z=11$$
$$4x+y+2z=12$$

上の式は次のような行列式で表すことができる．

$$\begin{bmatrix}1&2&4\\2&3&1\\4&1&2\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}17\\11\\12\end{bmatrix}$$

この行列を左からT，x，yとすると，T・x＝y，両辺にT^{-1}をかければ，T^{-1}・T・x＝T^{-1}・yとなるので，xはT^{-1}・yで求められることがわかる．「Tの逆行列T^{-1}を求めるにはどうするんだったっけ……」と迷わなくても，そこはTrueBASICの組み込み関数が処理してくれる（リスト2）．

リスト2のポイントになるのが，MAT文である．2，3行目のMAT READ文は，DATA文で定義された一連のデータを配列Tとyに代入する．4行目のMAT文は，組み込み関数INVを使って配列Tの逆行列を求め，それを配列Rに代入する．5行目で配列xを計算し，答を6行目のMAT PRINT文でプリントする．

この例からもわかるように，MAT文は配列操作に強力な威力を発揮する．

リスト2

```
dim T[3, 3], R[3, 3], x[3], y[3] !行列のサイズを定義
mat read T                        !Tの値を読み込む
mat read y                        !yの値を読み込む
mat R = inv(T)                    !Tの逆行列を計算する
mat x = R*y                       !xを求める
mat print x                       !答をプリントする
! T行列の内容
data 1, 2, 4
data 2, 3, 1
data 4, 1, 2
! xベクトルの内容
data 17, 11, 12
!
END
```

構造化プログラミング

一般に大きなプログラムは，いくつ

BASICプログラミング用語集

浮動小数点　非常に大きな数や小さな数を表現するのに適した科学的記数法に似た記数法．仮数部と指数部から構成されている．

かの小さなユニットに分割すると読みやすく，デバッグしやすく，変更しやすいものになる．TrueBASICにはプログラムを分割する手段として，ユーザー定義関数，サブルーチン，ピクチャという３つの構文が用意されている．このうちピクチャはグラフィックスでしか使用しない．

・サブルーチンと関数

サブルーチンは複数の文の集まりである．サブルーチンはSUB文ではじまり，END SUB文で終わる．メインプログラムからサブルーチンを呼び出すには，CALL文を使う．

関数もまた複数の文で構成されたブロックであるが，サブルーチンが「ある処理を行う」のに対して，関数はそれ自身が値をもつことができる，つまり「処理を行ってメインプログラムにある値を返す」という点が異なる．また，サブルーチンは引数の値を変更することができるが，関数は決して変更しないという点も違っている．

たとえば，２つの引数ＡとＢの値を入れ換えるという処理は，サブルーチンとして定義し，２つの引数ＡとＢの値を比較して，大きいほうを求めるという処理は関数として定義すると考えればよい．

TrueBASICでは，サブルーチンや関数のようにある処理を行う手続きのまとまりを「プロシージャ」と呼ぶ．

・ライブラリ

通常は，サブルーチンや関数はソースプログラムファイルに含まれるが，サブルーチンや関数をライブラリとしてファイルに保存しておけば，異なったプログラムからもそのサブルーチンや関数を利用できるようになる．ライブラリファイルは，必ずEXTERNAL文ではじまる．メインプログラムからライブラリに含まれる関数やサブルーチンを利用する場合は，LIBRARY文で使用するライブラリ名を指定しなければならない．

LIBRARY ”ライブラリ名”

また，ライブラリ中の関数を利用するときは，メインプログラム中でDECLARE DEF文で関数名を宣言する必要がある．これはABC(4)のような式が関数なのか，配列要素を示すものなのかを判別する際重要である．

DECLARE DEF ”外部関数名”

メインプログラムから外部関数や外部サブルーチンを使う場合，そこで利用される変数はすべてローカルな変数として取り扱われる．つまり，メインプログラムと外部関数，外部サブルーチンの変数名が重複しても，それらは互いに独立しているのである．ただし，その代償として，外部関数や外部サブルーチンで使用した値は一度その処理を終えた後は保証されない．ローカル変数の値を保持できるのは，後述するモジュールだけである．

・モジュール

TrueBASICでは，特殊なライブラリとしてモジュールという概念が導入されている．モジュールは，ライブラリの一種であるが，通常のライブラリと異なって，変数や配列，チャネルなどを所有することができる．

モジュールでは，変数をそのモジュール外からもアクセスできるようにしたり(PUBLIC)，モジュール内のすべてのプロシージャからアクセスできるようにしたり(SHARE)，そのモジュール内のみでしかアクセスできないプロシージャを作成（PRIVATE）することができる．

また，モジュールの先頭には，メインプログラムの実行前にモジュールの内容を初期化するプログラムが書ける．

リスト３は，変数one，two，threeを定義し，それぞれの値を１，２，３に初期化するモジュールである．このようにすれば，メインプログラム中からでも，ほかのライブラリからでも変数one，two，threeはそれぞれ値１，２，３をもった数値変数として取り扱うことができるようになる．

グラフィックス

・ウィンドウと座標系

TrueBASICには，グラフィックス処理機能がある．グラフィックス処理を行う場合には，まず画面（スクリーン）上にグラフィックスを描く範囲を設定する．これをウィンドウと呼ぶ．ウィンドウを開くには，OPEN文でウィンドウの範囲を指定してやればよい．TrueBASICでは画面の左下を

リスト３

```
MODULE  numbers
        PUBLIC  one, two, three
        LET  one = 1
        LET  two = 2
        LET  three = 3
END MODULE
```


BASICプログラミング用語集
ブートストラップ　短いプログラムで，目的の状態，たとえばOSの起動などを行う装置またはソフトウェア．

(0,0)，右上を(1,1)として座標を表示する(図3)．たとえば，画面の左上4分の1の大きさでウィンドウを開きたい場合は，次のようにする．#1はウィンドウを識別するためのチャネル番号である．

OPEN #1: SCREEN 0,.5,.5,1

また，ウィンドウはそれ自身で独自の座標系をもつことができる．デフォルトは，スクリーン全体の場合と同じく左下が(0,0)右上が(1,1)であるが，次の文ではウィンドウの座標系をx軸方向に0から2π，y軸方向に-1から1までと設定している．三角関数をグラフ化する場合などは，このような座標系を設定すると非常に処理が楽になるだろう．

SET WINDOW 0, 2* pi, -1, 1

点を描いたり，線を引いたりする処理は，設定した座標を使って行うことができる．それが実際に画面上のどの位置にあたるかという計算はTrueBASICにまかせればよい．

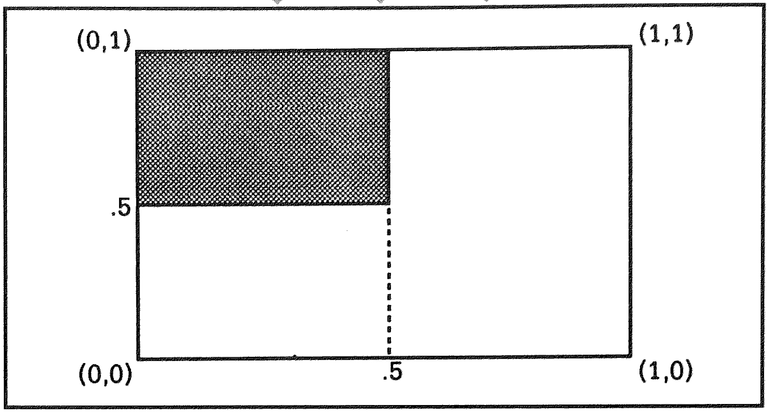
繰り返していうが，TrueBASICはハードウェア独立の言語であるから，スクリーンの解像度のように，特定の機種に依存した内容は知らなくてもプログラミングできるようになっているのである．

・ピクチャ

TrueBASICでは，グラフィックス専用のサブルーチンが用意されている．これをピクチャという．ピクチャはPICTURE文ではじまり，END PICTURE文で終わる．あるピクチャをメインプログラムの中から利用する場合は，DRAW文を使う（リスト4）．

DRAW ［ピクチャ名］

図3　グラフィックスの座標系



リスト4

```
! 円のピクチャを描きます。
!
PICTURE Circle(sides)

    FOR i = 0 to sides
       LET u = (i*2*Pi)/sides          ! Find next angle
       PLOT Cos(u), Sin(u);            ! Plot next segment
    NEXT i
    PLOT                               ! Turn off beam

END PICTURE

SET WINDOW -15, 15, -10, 10            ! Set the window
ASK MAX COLOR cmax                     ! Maximum colors

FOR j = 1 to 30                        ! Draw a bunch of circles
   SET COLOR Mod(j,cmax) + 1           ! In different colors
   DRAW Circle(j) with Scale(j/3)      ! Bigger and better
NEXT j
END
```

ピクチャを使うメリットは，座標系による図形の変換が簡単に行えるようになるという点である．ピクチャ自体はウィンドウを開いたり，座標系を決定したりはできないが，TrueBASICにはASK WINDOWという命令が用意されているので，これを使って現在カレントなウィンドウの座標系を知ることはできる．したがって，いったんある図形を描くというピクチャを作っておけば，ウィンドウの位置やサイズに関わらずこれを利用することもできる．

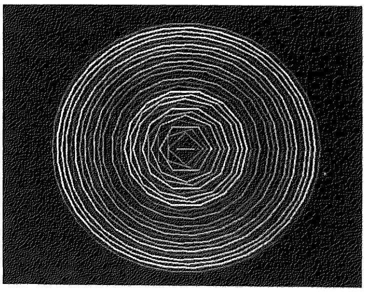


写真1　CIRCLE.TRUの実行結果

・その他の機能

そのほかにも，TrueBASICではグラフィックスを利用するために，表1のような各種の文が用意されている．

BASICプログラミング用語集
ブール代数　英国の数学者G.ブールが開拓したもので，論理的な命題の真偽関係を演算で処理するもの．命題算とも論理計算ともいう．

TrueBASICのセットアップ

では，次に実際の日本語TrueBASICの使い方を見ていくことにしよう．

TrueBASICをセットアップするには，添付されているセットアッププログラムを実行するだけでよい．

パッケージ内容

日本語TrueBASICのパッケージには，表2で示すマニュアルとプログラムディスク1枚が含まれている．リファレンスマニュアルは，J.G.KemenyとT.E.Kurtzの著書「TrueBASIC Version 3.0 Reference Manual」の翻訳書で「TrueBASICリファレンスマニュアル」のタイトルで一般の書店で市販されているのと同じものである．これは，コンピュータごとに製品パッケージがあっても，TrueBASICの言語仕様自体は1種類しかないという作成者の考え方に沿ったものである．日本語TrueBASICのエディタの操作方法やコマンドについてはユーザーズガイドを参照する．

インストール

プログラムディスクに含まれるTSETUP.EXEを実行すると，TrueBASICのセットアッププログラムが起動する．セットアッププログラムでは，ハードディスク用，2ドライブフロッピーディスク用，98NOTE用の3種類のインストール方法から選択するようになっている(画面1)．なお，ハードディスクにインストールする場合に必要なディスクの空き容量は約1MBである．

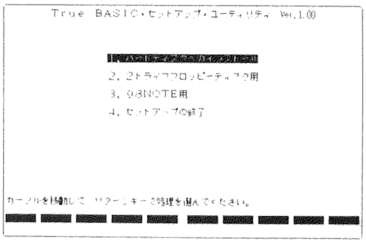
セットアッププログラムでは，バスマウスを使用するかどうか，またプリンタを使用する場合には，「NEC PC-PR系」「NEC NM系」「EPSON ESC/P系」から選択するようになっている．マウスを「使用する」を選ぶと，TrueBASIC用のマウスドライバがCONFIG.SYSファイルに組み込まれて，TrueBASICのエディタでマウスが使用できるようになる．プリンタ機種名を選択した場合には，対応するプリンタドライバがCONFIG.SYSファイルに組み込まれて，プログ

表1：グラフィックス文

ASK MAX CURSOR	カレントウィンドウの最大行数と最大桁数を取得する
ASK PIXELS	縦横のピクセル数を取得する
ASK SCREEN	カレントウィンドウの左右上下端をスクリーン座標で取得する
ASK TEXT JUSTIFY	文字列の水平と垂直の表示位置を取得する
ASK WINDOW	カレントウィンドウの左右上下端をウィンドウ座標で取得する
BOX AREA	指定した座標に長方形を描き，内部を塗りつぶす
BOX CIRCLE	指定した座標の長方形に内接する楕円を描く
BOX CLEAR	指定した座標の領域を消去する
BOX ELLIPSE	BOX CIRCLE 文に同じ
BOX KEEP	指定した座標の領域全体をピクセル形式で文字列に格納する
BOX LINES	指定した座標に長方形の枠を描く
BOX SHOW	文字列に格納されたイメージを長方形の領域に復元する
DRAW	ピクチャを描く
FLOOD	指定した2点を含む閉じた領域を塗りつぶす
GET POINT	プログラム実行中にグラフィックカーソルで座標を指定する
MAT PLOT	2次元の数値配列に格納されている点をプロットする
PLOT	前のPLOT文やPLOT LINES文がセミコロンで終わっていた場合，次のPLOT文やPLOT LINES文の始点とつながらないようにする
PLOT AREA	座標リストで定義した線を描き，囲まれた領域を塗りつぶす
PLOT LINES	座標リストで定義した点を結ぶ線を描く
PLOT POINTS	座標リストで定義した点を描く
PLOT TEXT	指定した座標に文字列を描く
SET TEXT JUSTIFY	PLOT TEXT 文で表示する文字列の水平と垂直の位置を設定する
SET WINDOW	ウィンドウ座標を設定する

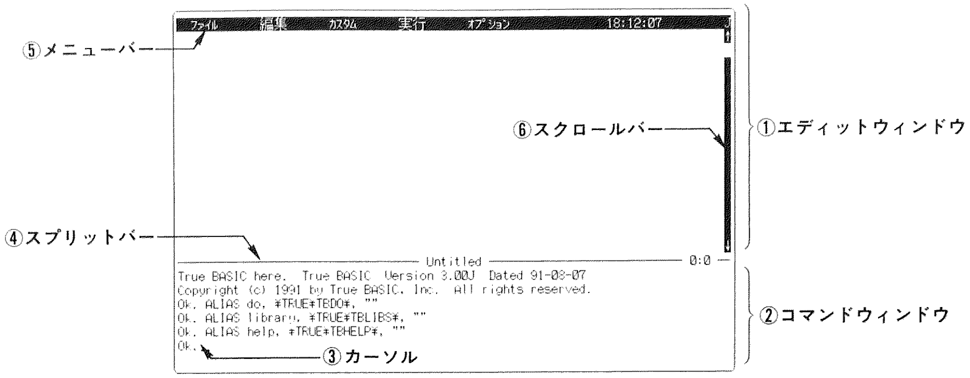
表2：日本語TrueBASICのマニュアル

マニュアル名	内容
セットアップマニュアル	日本語TrueBASICのインストールとセットアップについて
ユーザーズガイド	エディタの使い方など日本語TrueBASICの操作方法について
リファレンスマニュアル	TrueBASICの文法について
クイックリファレンス	キー操作，メニュー，コマンド，関数，文などの一覧



画面1　TrueBASICセットアッププログラム

BASICプログラミング用語集
ブリンク　フラッシングとも呼ばれ，ディスプレイ上の表示状態の一種で，カーソルや文字が点滅していること．



画面2　TrueBASIC 初期画面

ラムの実行画面のハードコピーがとれるようになる．

2ドライブフロッピーディスク用または98NOTE用のインストールを選択した場合は，さらにランタイムパッケージを転送するかどうかをたずねてくる．ランタイムパッケージは，TrueBASICを使って拡張子EXEの実行形式のファイルを作成する場合に必要である．

TrueBASICの操作環境

日本語TrueBASICの環境はエディタとインタプリタ，コンパイラ，デバッガが一体となったいわゆる統合型の開発環境である．マウスが利用できること，プルダウン方式のメニューがついていることなどは，QuickBASICなどほかの統合型の開発環境とよく似ているが，日本語TrueBASICの特徴はコマンド併用型であることだ．プログラムの呼び出しや保存，プログラムの実行などのおもな機能は，メニューを選択して実行できるのはもちろんだが，OLD，RUNなどのコマンドを入力して実行することも可能となっている．

TrueBASICの初期画面

TrueBASICを起動するには，MS-DOSのコマンドプロンプトからHELLOと入力する．ハードディスクから起動する場合には，あらかじめカレントディレクトリをTrueBASICをインストールしたディレクトリに変更してから入力する．

```
A>HELLO
```

TrueBASICが起動すると，画面2が表示される．ソースプログラムの入力・編集，プログラムの実行はすべてこの画面で行う．各部の名称と機能は次の通りである．

①エディットウィンドウ

TrueBASICの画面は，画面中央よりやや下で2分されている．上部はソースプログラムを入力・編集するウィンドウで，エディットウィンドウと呼ばれる．

②コマンドウィンドウ

TrueBASICにコマンドを与えたり，エラーメッセージやプログラムの実行結果を表示したりするウィンドウ．

③カーソル

TrueBASICでは，エディットウィンドウかコマンドウィンドウのどちらか一方にカーソルが表示される．カーソルが表示されているほうのウィンドウが入力可能となる．コマンドウィンドウからエディットウィンドウにカーソルを移動するには[f･1]キー，その逆は[f･2]キーを使用する．

④スプリットバー

エディットウィンドウとコマンドウィンドウの境界線．スプリットバーの位置は変更することもできる．バーの中央には現在エディットウィンドウで編集しているソースプログラム名が表示される（ただし，名称未定のときはUntitled）．また，バーの右側にはエディットウィンドウのカーソル位置が表示される．

⑤メニューバー

5種類のメニュータイトルとシステム時刻，現在選択されているモードが表示される．モード表示Jの場合は日本語，Eの場合は英語でメニューが表示される．

⑥スクロールバー

エディットウィンドウに一度に表示できないような長いプログラムで，マウスを使ってカーソルを移動させる場合に使用する．

コマンド

日本語TrueBASICでは，コマンド

ウィンドウからコマンドを入力することによって，ソースプログラムを編集したり，プログラムを実行したりすることができる．コマンドはコマンドウィンドウのOk.プロンプトの後に入力する．使用できるコマンドは表2のとおりである．なお，各コマンドはVIEWS KEYコマンドをのぞいて，最初の3文字だけの省略形を使うことができる．また，複数のコマンドをセミコロン(；)でつなげて記述すると，左から順に実行される．

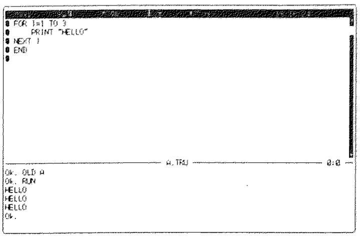
```
>Ok.REPLACE;RUN
```

のように入力すると，まずカレントなプログラムファイルをディスクに保存し，続いてそのプログラムを実行する．

メニューとファンクションキー

日本語TrueBASICでは，よく使われるコマンドや一部の編集機能をメニューとファンクションキーを使って実行することもできる．

マウスを使ってメニューを選択するには，マウスカーソルをメニューバーの選択したい機能に合わせて左ボタンを押す．プルダウンメニューが開かれるので左ボタンを押したままマウスを



画面3　RUNコマンドを実行したところ

下方向に移動させて，選択したい機能が反転表示されたら，そこで左ボタンを離せばよい．キーボードを使う場合は，まず[GRPH]キーを押して「ファイル」メニューを表示する．あとは[←][→][↑][↓]キーを使って選択したいメニューを反転表示させてからリタ

表3-1：日本語TrueBASICで使用できるコマンド一覧(1)

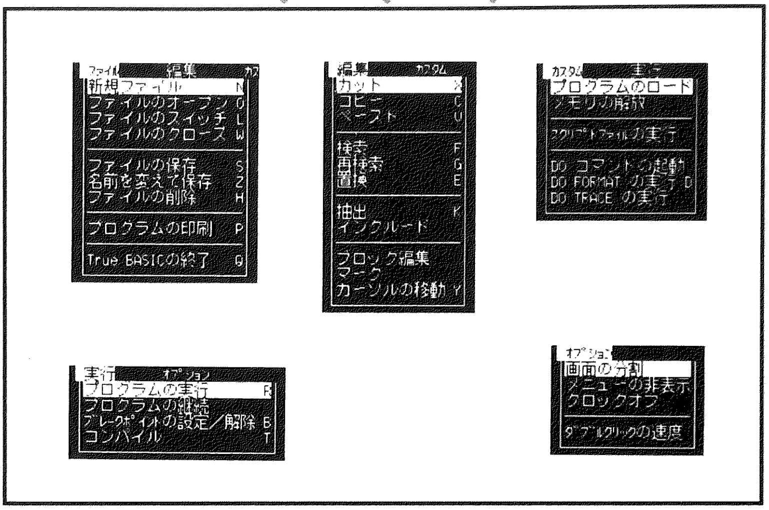
［プログラムファイル操作］	
CD	カレントディレクトリを変更する
CLOSE	プログラムをクローズする
ENTER	CDと同じ
NEW	エディットウィンドウのプログラムを消去する
OLD	ディスクに保存されているソースプログラムを呼び出して，エディットウィンドウに表示する
RENAME	プログラム名を変更する
REPLACE	エディットウィンドウに表示されているプログラムをディスクに上書きして保存する
SAVE	エディットウィンドウに表示されているプログラムをディスクに保存する
SWITCH	プログラムのスイッチ処理の可／不可の切り換え（引数なし） 編集対象のプログラムを切り換える（引数あり）
UNSAVE	ディスクに保存されているファイルを削除する
［実行］	
COMPILE	プログラムをコンパイルする
RUN	プログラムを実行する
［編集］	
CHANGE	文字列を一括置換する
COPY	マークした行を複写する
DELETE	マークした行を削除する
EDIT	マークした行を編集対象とする
FIND	文字列を検索する
INCLUDE	カーソル位置へ保存されているプログラムを挿入する
KEEP	マークした行を抽出する
LOCATE	指定した文字列を含む行を表示する
MARK	マークをつける
MOVE	マークした行を移動する
TRY	文字列を逐次置換する
TO	指定した行番号，関数，サブルーチンにカーソルを移動する

BASICプログラミング用語集
フロントエンドプロセッサ　コンピュータとユーザーの間で前処理や別の処理を行うことのできるプロセッサのこと．おもに日本語入力処理をいう．

ーンキーを押す．メニューの右横にアルファベットが表示されているものについては，アルファベットに対応したキーを押すことでメニューを選択できる．

また，ファンクションキーにも，よく使われる機能が割り当てられている．SPLITコマンドを使うと，ファンクションキーに割り当てられた機能を画面上のスプリットバーの下に表示することができる(画面4)．

図4　メニュー一覧



エディタ

日本語TrueBASICのエディタは，

表3-2：日本語TrueBASICで使用できるコマンド一覧(2)

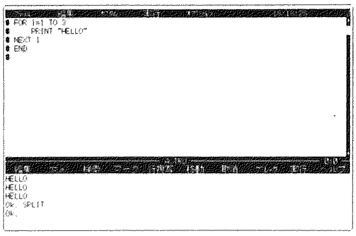
［デバッグ］	
BREAK	ブレークポイントを設定する
CONTINUE	停止したプログラムを再開する
［情報照会］	
FILES	ディスク中のプログラムファイルのリストを表示する（引数なし） 指定した種類のファイルのリストを表示する（引数あり）
HELP	ヘルプ画面を表示する
INFO	現在の TrueBASIC の環境情報を表示する
［その他］	
ALIAS	ALIAS コマンドで指定した内容を表示する（引数なし） ファイル種類による指定内容を表示する（引数1つ） 指定した種類のファイルが含まれるディレクトリを指定する（引数2つ以上）
BYE	TrueBASIC を終了する
COLORS	エディタ各部の色の設定をファイルに保存・呼び出しする
DO	プリプロセッサを実行する
ECHO	コマンドの実行結果をプリンタやファイルに出力する
FORGET	メモリを解放する
KEY	キー操作をあるキーに定義する（引数なし） 定義したキー操作をファイルに保存・呼出する（引数あり）
LANGUAGE	メニューバー，ファンクションキー，メッセージの表示を日本語（LANGUAGE　JAPAN）または英語（LANGUAGE ENGLISH）に切り換える
LIST	プログラムを印刷する
LOAD	ライブラリファイルをロードする
MENUS	メニューバー，スクロールバー，マウスの表示を消す
MODE	崩れた画面を回復する
NOLET	プログラム中の LET 文を不要にする
SCRIPT	スクリプトファイルを実行する
SPLIT	ファンクションキーの機能を画面に表示する（引数なし） スプリットバーの位置を変更する（引数あり）
VIEW KEYS	定義されているキーと定義内容を表示する

複数のファイルを編集できるマルチファイルエディタである．ファイルは，ディスク上に作業ファイルを作成するための空き領域がある限り，いくつでもオープンできる．エディタには，行単位でのコピー，カット，ペースト，文字列の検索，置換，特定のサブルーチンだけを編集の対象とする機能など，プログラムの編集に必要なひと通りの機能がそろっている．

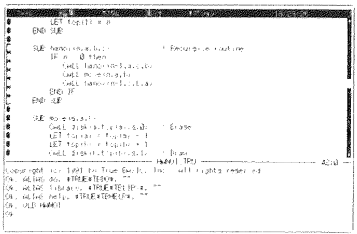
特定の行のまとまりをコピーしたり削除したりするには，その行にマークをつける．行にマークをつけるには，指定したい行のまとまりの先頭行にカーソルを合わせて

(A) [f･4]キーを押す
(B) MARKコマンドを入力する
(C)「編集」メニューから「マーク」を選択する

のいずれかの操作を行う．次に行のまとまりの最終行にカーソルを移動してもう一度(A)～(C)のいずれかの操作を行う．一連の操作で「マーク」された行は，行の左端にある「ラインタグ」が反転表示される(画面５)．マークされた行をコピーするには，コピー先にカーソルを移動して[f･5]キーをおせばよいし，削除するには[DEL]キーを押せばよい．また，[>][<][!]キーを使えば，行のまとまりごと１カラム右・左に移動させたり，行のまとまりごと注釈文にしたりすることもできる．



画面４　SPLITコマンド実行する



画面５　ラインタグが反転表示される

表４：ファンクションキーの機能

キー	機能
[f･1]（編集）	カーソルをエディットウィンドウに移動する
[f･2]（コマンド）	カーソルをコマンドウィンドウに移動する
[f･3]（検索）	エディットウィンドウで文字列を検索する
[f･4]（マーク）	エディットウィンドウで行にマークをつける
[f･5]（行複写）	（エディット）マークされた行を複写する （コマンド）他の行の内容をコマンド行に複写する
[f･6]（移動）	（エディット）マークされた行を移動する （コマンド）[f･5]に同じ
[f･7]（取消）	移動，複写，削除の操作を取り消す
[f･8]（ブレーク）	エディットウィンドウにブレークポイントを設定・解除する
[f･9]（実行）	プログラムを実行する
[f･10]（ヘルプ）	ヘルプ画面を表示する
[SHIFT]+[f･1]	（エディット）垂直に上のタブに移動する
[SHIFT]+[f･2]	（エディット）垂直に下のタブに移動する
[SHIFT]+[f･3]	（エディット）しおりまたはブレークポイントを検索する
[SHIFT]+[f･4]	（エディット）しおりを設定・解除する
[SHIFT]+[f･9]	一時停止したプログラムの実行を継続する

このエディタのユニークな点は，ひとつのファイルしか編集できないように機能を制限することが可能なことだ．引数なしでSWITCHコマンドを使用すると，それ以後もう一度SWITCHコマンドを入力するまで複数ファイルの取扱いができなくなるのである．これは，初心者に対する配慮から出た制限であろう．ひとつのファイルしか編集できないという制限をつけておけば，プログラミングの学習者がよけいな混乱を招くおそれは少なくなる．後述するように，日本語TrueBASICには環境をカスタマイズする機能が備わっているので，あらかじめ指導者がそのように設定しておけば，学習者は自分でSWITCHコマンドを入力しなくても，起動するだけで機能の制限されたエディタを使うようにできる．

オンラインヘルプ

日本語TrueBASICでは，コマンドや関数などについての説明（ヘルプ）を画面上に表示させることができる．ヘルプを表示するには，HELPコマンドを使用するか，[f･10]キーを押す．

・HELPコマンド

コマンドウィンドウでHELP（調べたいコマンド・関数名）と入力する．たとえば，RUNコマンドの説明を見るには次のようにする．

```
Ok.HELP RUN
```

・[f･10]キー

[f･10]キーを押すと画面中央にコマンド・関数名を入力するボックスが表示される．ここでコマンド・関数名などを入力してリターンキーをおすと，ヘルプ画面が表示される．

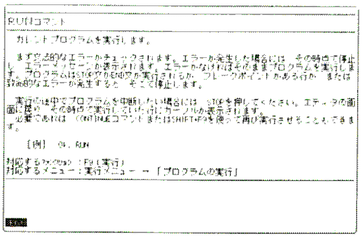
BASICプログラミング用語集
マトリクス　２次元配列のこと．グラフィックス画面などで座標を指定するのに使用される．

プログラムの実行

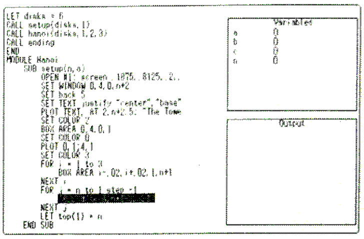
エディタを使って作成したプログラムはただちに実行することができる．実行の方法は次の3通りがある．

(A)[f･9]キーを押す．
(B)コマンドウィンドウでRUNコマンドを入力する．
(C)「実行」メニューから「プログラムの実行」を選択する．

実際に簡単なプログラムを作成して実行するまでの基本サイクルは，画面7～10のようになる．④の操作は，もちろん上記(B)(C)でも構わない．文法上のエラーがあればコマンドウィンドウにエラーメッセージが表示される（画面10）．



画面6　オンラインヘルプ



画面12　プログラムのトレース

表5：トレースのオプション

オプション	内容
slow	ゆっくり実行する
fast	速く実行する
step	キーを押すごとに1行ずつ実行する
break（条件）	fast で実行し，変数がある条件を満たすと step に切り換わる

プログラムのデバッグ

日本語TrueBASICには，作成したプログラムをデバッグするための機能として，プログラムのトレース機能，変数のトレース機能，ブレークポイントの設定機能がある．また，TrueBASICそのものの仕様の中に，プログラムをデバッグするための文として，DEBUG文，BREAK文，TRACE文の3種類の文が用意されている．

・トレース

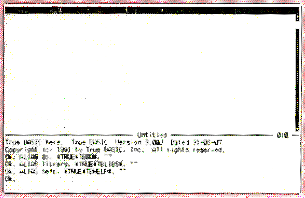
日本語TrueBAISCのトレース機能は，プリプロセッサの形で実現されている．コマンドウィンドウからDOコマンドでTRACEプリプロセッサを呼び出す．このとき，表5のオプションが指定できる．デフォルト値はstepである．また，変数名を8つまで指定してその内容を監視することもできる．変数x，yとS$の内容を表示し，slowでトレースするには次のように入力する．

```
Ok.DO trace, slow (x,y.S$)
```

・ブレークポイントの設定

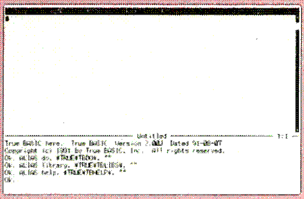
ブレークポイントは，設定したい行にカーソルをあわせて，[f･8]キーを押すだけで設定できる．[f･9]キーを押してプログラムを実行すると，ブレークポイントを設定した行の直前でプログラムが停止する．日本語TrueBASICでは，コマンドウィンドウで直接BASICの文を入力することができるので，停止した時点での変数

①日本語TrueBASICを起動する（画面7）



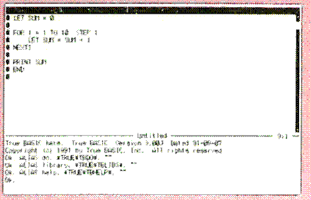
↓

②[f･1]キーを押して，カーソルをエディットウィンドウに移動する（画面8）



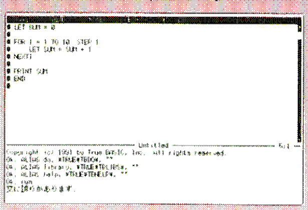
↓

③プログラムを入力する（画面9）



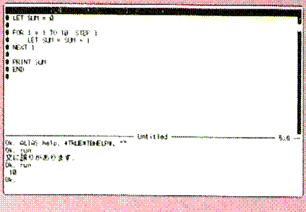
→

④[f･9]キーで実行，エラーがあれば表示される（画面10）



→

⑤エラーを修正して，正常終了（画面11）



の内容を調べることができる．

・**デバッグ文の使用**

ソースプログラムに直接TRACE ON，TRACE OFF，BREAKなどと記述して，トレースを出力したり，ブレークポイントを設定することも可能である．デバッグ文の使用を開始するときには，DEBUG ON文，これを終了するときにはDEBUG OFF文を記述する．ソースプログラム中に多数のTRACE文を埋め込んでしまっても，DEBUG ON，DEBUG OFF文を使えばこれらのTRACEをやめることも簡単にできる．

EXEファイルの作成

これまで見てきたのはインタプリタの使い方であるが，TrueBASICでは，ランタイムパッケージを使って単独で実行できるEXEファイルを作ることもできる．この作業をバインドという．バインドは次の手順で行う．

①「実行メニュー」から「コンパイル」を選択する．コンパイルが正常に終了すると，拡張子TRCのファイルが作られる．

②BYEコマンドでTrueBASICを

Interview 日本語版の開発元㈱クレオに聞く

株式会社クレオ　システムプロダクト事業本部
パソコンプロダクト営業部商品開発部技術開発課　梅田明由

―パソコン版のTrueBASICはどういった経緯で登場することになったのでしょう

ダートマス大学では，1964年にBASICが開発されて以来，現在のTrueBASICの元になったBASICが使われてきました．大学内には計算センターがあって，そのタイムシェアリングと共存できる形で，一般のパソコン市場で広まったBASICとは別の途を歩みながら育っていったのです．

TrueBASICがはじめてパソコンに載ったのは1984年です．1974年にFULL BASICのANSI規格が制定されたのですが，このとき，BASICの開発者のひとりであるKurtz博士が委員長として参加していました．FULL BASICの制定にあたって，BASICはどうあるべきかという議論が重ねられたわけですが，その仕様を満たす処理系が実際にはまだ存在していなかったため，当時のダートマス大学版のBASICをもとに，パソコン上でもUNIX上でも動く処理系としてTrueBASICが作られました．

「Back to BASIC」の中でも触れられているように，当時のパソコンBASICにおけるさまざまな問題点を解消するために，またもうひとつにはANSIのFULL BASICの規格を満たす処理系を作るという２つの理由から，TrueBASICのパソコン版が生まれたと考えていただければよいでしょう．

当時はBASICがもてはやされて，爆発的に広まっていた時期にあたります．そのしばらく後にどんどん批判されて衰退していくことになるのですが，当時はBASIC，それも開発者が理想としていた形からはかけ離れたものが市場に流布されていたというのが，かなり強いきっかけになっているのだと思います．

―1984年に登場したときは，MS-DOS上の処理系だったのですか

おそらくいちばん最初はIBMではないと思います．はっきりとそう聞いたわけではないのですが，TrueBASICのアドレスのとり方は，フラットなメモリ空間を想定していて，セグメントやオフセットという概念は存在しません．そういう意味では68000やUNIXに通ずる考え方ですね．おそらくIBM PC版が出るときにも試作的にUNIXなどの大きなコンピュータで作ってみたのではないでしょうか．その辺りのことはあくまで類推にすぎませんが．

―アメリカの大学では，TrueBASICはかなり広く使われているそうですね．数でいうとUNIXベースよりはDOSベースのものが多いのでしょうか

UNIXベースのTrueBASICは，グラフィックスが使えません（ただし，現在は使えるものもあります）．おそらくサイトライセンス契約は，MacintoshとIBMがメインで行われていると思います．

―TrueBASICのツールキットは，どういう位置づけになるのでしょう

「BASICというのはシンプルでスマートであるべきだ」ということから，私どもではよけいな機能追加はしていません．ある特別な処理を行いたいとか，グラフを描きたいというときには，BASICそのものに機能を追加するのではなくて，必要な方にはツールキットという独立した製品を購入していただいて，その機能を利用していただくという方法をとっております．

ですから，数値をグラフ化したいとか，データを直に見てみたいという要求はあるが，プログラムの知識もないし，それほどの時間もかけられないという方に，すぐに利用していただけるアプリケーション的なサブルーチン群を用意しています．TrueBASICは，プログラムの学習をするためのツールだけでなく，プログラムが必要だ

BASICプログラミング用語集
文字列　数字や記号，アルファベットなど文字として扱うデータの配列．BASICでは文字列というデータ型が用意されていて，配列として扱う場合は少ない．

終了する.

③MS-DOSのコマンドラインから, BINDと入力し, 次に①で作られたファイル名（拡張子は省略可能）を指定する.

（例） A>BIND HANOI

④ BINDコマンドが正常に終了すると“xxxx.EXEファイルを出力しました”というメッセージが表示される.

操作環境のカスタマイズ

日本語TrueBASICの操作環境は, ユーザーが好みにあわせて変更できるようになっている.

・色の変更

ウィンドウの文字色, メニューバーとスクロールバーの色はそれぞれ[CTRL]+[F], [CTRL]+[G]キーを押すたびに黒→赤→マゼンタ→緑→シアン→黄色→白の順で変わっていく. 好みの設定はCOLORSコマンドでファイルに保存したり, 呼び出したりできる.

色の保存

COLORS TO [ファイル名]

色の呼び出し

Interview

けれどしない, できない, 時間がないという方にBASICを「利用」していただく, いわばBASICの世界を広げるものだと考えています.

ーツールキットの発売状況はどうなっているのでしょう

現在は「ビジネス用グラフィックスツールキット」と「数学者用ツールキット」が発売されていますが, 今年じゅうに「科学技術用グラフィックスツールキット」「統計用グラフィックスツールキット」「並べ替え・検索用ツールキット」「文字列操作ツールキット」「3次元グラフィックスツールキット」の5つが発売されます.

ツールキットの内容を見ていただければわかりますが, これを使えば, まだ理論的な裏付けの理解ができない中学生くらいの方でも, プログラムを勉強したり, グラフを描いたり, アニメで遊んでみたりということが簡単にできます.

ーかなり教育用ということを意識されているようですが, ツールキットと構成の関連はあるのですか

とくに教育用を意識してツールキットを構成したわけではありません. もとがアメリカで発売されているもので, 分野別のラインアップにはなっています. 基本的にはアメリカで発売されている路線のままでいく予定です.

ー教育用としては, どのレベルをターゲットにしているのですか

ツールキットを使うという前提では, 大学や工業高校など, 実験を行ってデータを見る, 情報を加工する必要のある分野を考えています. もっと初等の段階「プログラムとは何か」「コンピュータとは何か」というレベルでも, 言語としてTrueBASICを使っていただきたいと考えています.

ー現在, 中学校での情報教育が問題になっていますね. 教える側へのトレーニングなどについてはどのようにお考えですか

平成5年からは, 中学校のカリキュラムに情報処理が入ってきます. そこで各地の教育センターでは, 学校の先生を集めて先生用のトレーニングを行っていますので, 私どもでもそちらをまわってTrueBASICの内容説明などにうかがっています. 教育関係の方からは高い評価をいただいています. マイクロソフト系のBASICの場合ですと, データ型が複雑であるとか, 自分の入力した文字が勝手に大文字になってしまうとか, 大きな数字を入力すると, 突然指数表示になったりだとか, 場合によっては生徒が混乱してしまうような点がありますが, TrueBASICではそういうことはありません.

ー今後「開発支援ツールキット」の販売も予定されているそうですが, ツールキットを使って作成したアプリケーションプログラムを販売したり, 学校内で配布したりすることはできるのでしょうか

ツールキットそのものを販売するというのは困りますが, ツールキットを使って作られたアプリケーションは自由に配布していただいて構いません.

ーツールキットはすべてTrueBASICで書かれているのでしょうか

開発支援用, 通信サポート, 数学者用の一部などにアセンブラでコーディングされているものもあります. これらについては, ソースプログラムは公開しておりません. 文字列操作などはTrueBASICで書かれておりますし, ソースプログラムも付属しています. 逆にこういったツールキットを利用したり, ほかのメーカーさんからも各種のツールキットが販売されるようになってくると, TrueBASICの活用範囲もますます広がってくるのではないでしょうか.

COLORS FROM ［ファイル名］

・英語／日本語表示の切り換え

日本語TrueBASICには，日本語モードと英語モードがある．現在のモードはメニューバーの右端にJ（日本語）またはE（英語）と表示される．デフォルトは日本語モードであるが，英語モードにするとメニュー，ファンクションキー，メッセージなどの表示がすべて英語になる．モードを変更するにはLANGUAGEコマンドを使用する．

日本語→英語

LANGUAGE ENGLISH

英語→日本語

LANGUAGE JAPANESE

・キー操作の定義

ユーザーは[CTRL]+[A]～[CTRL]+[Z]までの26種類のキーに任意のキー操作を定義することができる．たとえば，[CTRL]+[R]にREPLACEコマンドを定義するにはKEYコマンドを使用する．

①コマンドウィンドウでKEYと入力する．

②「定義するキーを押してください」と表示されるので[CTRL]キーと[R]キーを同時に押す．

③「定義内容：」と表示されるので，「REPLACE」と入力してリターンキーを押す．リターンキーは「*」で表示される．

④[ESC]キーを押して定義終了．

以後は[CTRL]+[R]キーを押すと「REPLACE（リターン）」と入力したのと同じことになる．また，キーの定義はファイルに保存・呼び出しすることも可能である．

キー定義の保存

KEY TO ［ファイル名］

キー定義の呼び出し

KEY FROM ［ファイル名］

現在定義されているKEYを知りたい場合は，VIEW KEYSコマンドを使う．

・スタートアップファイルの設定

日本語TrueBASICは，起動時にまずカレントディレクトリにあるSTARTUP.TRUファイルの内容を実行する．これはMS-DOSにおけるAUTOEXEC.BATファイルと同じような役割を果たす．色の定義や，キー操作の定義などを呼び出すコマンドを記述しておけば，毎回自分で設定しなおさなくとも，自動的に好みの環境を構築することが可能である．

表6：ツールキット

名称	内容	発売時期	価格
3次元グラフィックスツールキット	簡単な3次元の図形を表示．遠近法の利用も可能．	11月上旬	18,000円
並べ替え・検索用ツールキット	数値配列，文字列配列の並べ替えや検索を行う．クイックソート，ヒープソート，選択ソートなどツールも含む．	11月下旬	15,000円
科学技術用グラフィックスツールキット	通常のグラフ，対数，片対数，極座標グラフや，最小2乗法，スプライン，ベジェ曲線のグラフなどを描く．	11月上旬	25,000円
ビジネス用グラフィックスツールキット	棒グラフ，ヒストグラムなど各種のビジネスチャートを描く．	発売中	18,000円
データ入力フォーム管理ツールキット	フルスクリーンの入力形式をサポート．フィールドの反転表示や保護，カーソル移動などが行える．	未定	未定
数学者用ツールキット	複素数計算や，高精度（100,000桁）の実数計算などを行う．	発売中	25,000円
通信サポート用ツールキット	1つまたは2つのシリアルポートにアクセスする．19,2000bpsの速度で読み書き可能．	未定	未定
開発支援ツールキット	BIOSやDOS割り込みに対応．画面制御，ポップアップメニューやディスク操作などもサポート．	未定	未定
文字列操作ツールキット	パターンマッチング関数のスキャンと計算，連想メモリ，文章への空白追加，行末揃えなどの文書編集．	11月下旬	15,000円
統計用グラフィックスツールキット	統計用のグラフ処理や平均値，中央値，ウインザライズ平均，線形回帰，分散分析，多重回帰などの統計計算．	12月上旬	25,000円

BASICプログラミング用語集
リダイレクト　プログラムが入出力を行うデバイスとして，デフォルト以外のデバイスを指定すること．

TrueBASICの機能拡張

TrueBASICには，機能拡張のための手段が用意されている．ひとつはLOADコマンドによって，特定目的のためにTrueBASICそのものを機能拡張する方法であり，もうひとつは別売の各種ツールキットを利用する方法である．

LOADコマンドによる機能拡張

TrueBASICでは，サブルーチンや関数を別ファイルにライブラリやモジュールとして保存し，プログラムから利用できることはすでに説明した．LOADコマンドを使ってライブラリやモジュールを読み込んでメモリ上におき，実質的にTrueBASICシステムの一部として利用することもできるようになっている．

たとえば，関数LENGTHとWIDTHを含むライブラリをMEASUREとする．ふつうはメインプログラムでLENGTHやWIDTHを使うには，次の3つの文が必要である．

```
LIBRARY "MEASURE"
DECLARE DEF LENGTH
DECLARE DEF WIDTH
```

ところが，あらかじめコマンドウィンドウからMEASUREをロードしておけば，これらの文がなくても，メインプログラムからLENGTH関数やWIDTH関数を使えるようになる．また，いったんロードしておけば，ライブラリが常駐した状態になるので，読み込みやリンクの手間がなくインタプリタの実行がスピードアップする．

特定のライブラリを自分用に作成して，TrueBASICのスタートアップファイルで自動的にLOADするよう設定しておけば，いつでも専用の拡張機能を組み込んだTrueBASICを使うことができる．

ツールキットによる機能拡張

(株) クレオでは，TrueBASICの各種ツールキットを順次発売する予定である．ツールキットとは，分野別のサブルーチンやユーティリティを集めたもので，現在表6にあげたものが用意されている．提供されるサブルーチンやユーティリティの多くは，それ自体がTrueBASICで書かれており，ソースプログラムも添付されているので，内容を変更して自分なりに書き換えることもできる．内容的にはかなり高機能な処理のできるものが多く，冒頭であげた原則の3番目「高度な機能は追加機能であること，したがって，もしお金がかかるのなら初心者ではなく熟練者が負担すること」が，まさにその通りの形で実現されている．

COLUMN

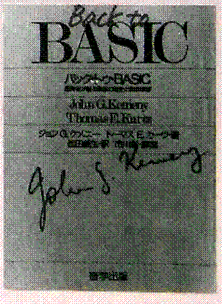
Back to BASIC
―Kemeny & Kurtzが語るBASICの歴史と開発思想

BASIC言語の設計者として知られるJ.G.KemenyとT.E.Kurtzが一般向けとしてははじめてBASICの歴史と，開発の思想について語った1冊．

1963年のダートマス大学の決定「すべての在学生にコンピュータを教え，使いこなせるようにする」を実現するために，2人がどのようにBASIC言語を作っていったかという開発の経緯，文字列処理やINPUT文などいまでは当たり前の機能が，実は後からつけ加えられたものであること，その後，グラフィックス処理や構造化プログラミングがどのようにとりいれられていったか，そしてパソコンの普及とともに，本来のBASICとは異なったさまざまなストリートBASICが登場して，2人が新たなBASIC言語（後にTrueBASICとなる）を作る決意をするにいたった経緯などが語られている．

本文中では，「何がいけなかったのか？」と題した1章を設けて従来のBASICの問題点を分析したり，その問題点を克服したTrueBASICがLogo，Pascal，FORTRAN，COBOL，Adaなど他のプログラミング言語と比較して教育用として圧倒的に適した言語であるとも述べられている．

また，TrueBASICによるサンプルソースも多数掲載されている．



「バック・トゥ・BASIC」(啓学出版，2,000円)

レジスタ　プロセッサが命令を実行するときレジスタと呼ばれるCPU内部のメモリを参照する．レジスタは外部のメモリに比べ高速に動作する．

パソコン博物館 XII

はじめての16ビット機
MULTI-16

三菱電機 MULTI-16

井戸禎光

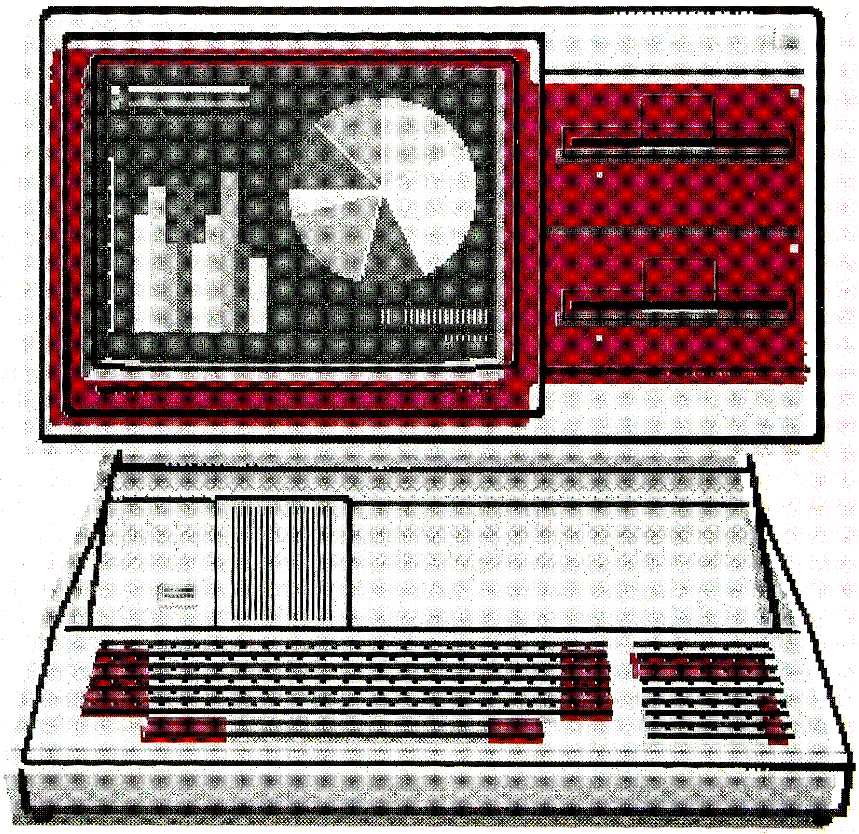


Illustration
TOSHIHITO ITO

日本ではじめての16ビット CPU を採用したパソコンは，1981年，16ビット変換期にビジネスユースを狙ってはじめてパソコンビジネスに参入した三菱電機の MULTI-16であった．1981年は，米国においても IBM 社が IBM PC を発表した年である．

実際，日本における 8 ビット期の黄金時代は1980年から1982年の 3 年間であったといえる．この 3 年の間に 8 ビット機の基本性能が確立した．別のいい方をすると技術の円熟した機械が各社から次々に発表されている．一般的には英数カタカナが表示でき，言語は BASIC であった．

このような状況において発表されたのが，三菱電機の MULTI-16である．他社と同じような路線で 8 ビットの完成度の高い機械を出さずに，あえ

て16ビットとしたのは，やはり永年制御用を含めてミニコンなどで培ってきた技術の発露であろうか．ほかの多くの16ビット機が発表されるのは1983年から1985年にかけての3年間である．

CPU はIBM と同じくインテルの8088，RAM はさすが16ビットということで256KB，ディスプレイは8色カラーで640×400ドット，外部メモリとして5.25インチタイプの320KB FDD を2基内蔵といったところまで，現在のパソコンと比較してもあまり遜色ない内容である．ただOS がディジタルリサーチ社のCP/M-86というところが異なっている．

価格は123万円で，この価格はアマチュア向けにはちょっと手のでないものであろうが，Multiplan などを使うビジネスユースには，工業用品並の高信頼性を誇っているのが特徴の三菱電機製では，やむを得ないというところであろうか．

当時の8ビット／16ビット論争は，今日の16ビット／32ビットの変換期と似ている面がある．しかし，当時は8ビットはBASIC，16ビット機はDOS といったぐあいにOS がはっきり異なっていた．これはフロッピーディスクが設計進歩によって価格低下期に入るのと一致したために可能になったのであろう．

BASIC は8ビットでも十分実用になるという意味で優れた言語であるが，企業におけるいわゆる業務用といわれるプログラムは，非常に大きく，FORTRAN，COBOL のような言語で開発されたものが多い．このような業務用アプリケーションプログラムを走らせるには，プログラムに使えるメモリ空間が大きく，漢字の表示および印字が可能，数種類の言語が使用可能などの条件が必要になり，これが16ビット機を必要としていたのである．

当然，外部記憶装置もカセットテープではなくフロッピーディスクとなり，ハードディスクも接続可能となっている．漢字の種類は500字が標準で，128KB を加えることによって4000字が表示できるようになっていた．

構造的にはif800と同じような一体構造をとっていたがプリンタは別であった．キーボードは94キーで8041を使ってスキャンしている．

CPU の8088はデータバスが8ビットであるだけが異なっているが，8086とソフトウェア的には互換性がある．このCPU に4.44MHz のクロックを加え，バスサイクルは1.125～1.35μ sec としており，外部信号で延長することが可能なようになっている．CRT 表示用のドットクロックは13.3MHz，キャラクタクロックは1.66MHz にしてある．水平走査は70.35Hz としている．メモリはアクセスタイム160nsec の64K DRAM を使用してリード／ライトのアクセスタイムは30nsec としている．

さてこのような高性能マシンが出たわけであるが，その後のアプリケーションソフトの充実，代理店網などの販路の組織化に十分な資源の投入がされず，他機の追随を可能にし，マーケットがプルからプッシュに変わると，あまり市場に影響力をもたなくなったことは残念である．

しかし，本機は16ビットパソコンの基本的な回路構成を決め，そのひとつのスタイルを先取りした点で画期的な作品といえよう．現在はAX 機でリーダーシップをとっている企業のひとつであるが，その起源を探るとこのようないきさつがあるようである．

● MULTI-16のバス・インターフェース信号

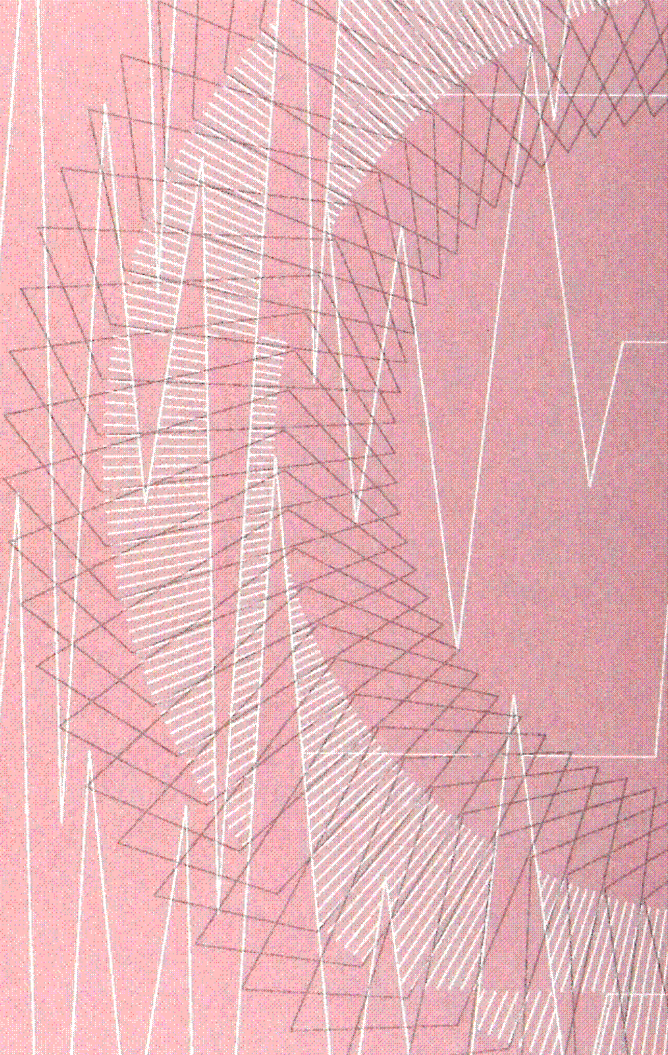
ピン番号	信号の名称	ピン番号	信号の名称
1	GND	2	+5V
3	GND	4	* Reserved
5	* PBOREL	6	* PADR13L
7	* PADR12L	8	* PADR11L
9	* PADR10L	10	* PADR0FL
11	* PADR0EL	12	* PADR0DL
13	* PADR0CL	14	* PADR0BL
15	* PADR0AL	16	* PADR09L
17	* PADR08L	18	PADR07L
19	PADR06L	20	PADR05L
21	PADR04L	22	PADR03L
23	PADR02L	24	PADR01L
25	PADR00L	26	* Reserved
27	* PMWICL	28	* PMRDCL
29	PIOWCL	30	PIORCL
31	PXACKH	32	PSELL
33	PINTOL	34	PANEL
35	* PTCTL	36	PINITD
37	* PCCLKL	38	PDATA7L
39	PDATA6L	40	PDATA5L
41	PDATA4L	42	PDATA3L
43	PDATA2L	44	PDATA1L
45	PDATA0L	46	* PINTAL
47	* PINTRL	48	* PDRQ1L
49	* PNMIL	50	* PDACK1L
51	* PDRQ2L	52	* PDRQ3L
53	* PDACK2L	54	* PDACK3L
55	* BRQGTL	56	-12V
57	+12V	58	GND
59	+5V	60	GND

*：外部では使えない信号

祐安重夫

プログラマの道具学

第12回　電子メール



パソコン通信サービスは中央にホスト・コンピュータがあり，そこをとおして電子メールや電子掲示板による情報のやりとりをおこなう．一般には電子メールをおくったりうけとったりすることのできる相手は，そのサービスに加入しているメンバーにかぎられている．

現在のように大規模なサービスから草の根BBSまで，複数のネットワークが並列して存在していると，電子メールをおくりたい相手がおなじサービスを利用していないと，おくることができないということになる．

だからといってパソコン通信を，中央集権的なひとつのシステムに統一しようとかんがえるのは，非常に危険なことである．自由でちいさなメディアを，誰もがつくり利用することができるという状態は，のぞましいことである．

たとえば電話をかんがえてみよう．アメリカでは電話は最初から独占事業ではなかったので，複数の電話会社が存在し競争をおこなっていた．だからといってある電話会社の回線に接続された電話機から他の電話会社の電話をよびだすことができなければ，不便なことはいうまでもない．そこで電話会社は相互に回線を接続し，中継サービスをおこなってきたのである．日本でもNTTが民営化され，電話事業が自由化されるとともに，おなじことがおこなわれるようになってきている．

パソコン通信の場合も，ホスト同士を接続し情報の交換をおこなうことで，電子メールを転送するといったサービスが，一部でおこなわれるようになってきており，ゲートウェイなどという言葉でよばれている．

これとはすこしことなるアプローチで，1984年頃からJUNETとよばれるネットワークの実験がはじまっている．大学や研究所のようなところでは，大型コンピュータやUnixベースのミニコンピュータなどに，多数のユーザが接続するというかたちの利用をおこなっていた．それらのシステムには電子メール機能もふくまれており，ローカルな情報の交換につかっているところもあった．

Illustration
Mikio Sasaki

JUNETはそれらのホストを電話回線で相互に接続し，電子メールのやりとりができるようにしたのである．UnixシステムにくみこまれていたUUCPという相互接続のプロトコルを利用し，直接接続されていないコンピュータとも，バケツリレー方式で中継しながら情報のやりとりをおこなうというものである．現在ではJUNETは実用的なシステムに発展し，日本全国の大学や研究機関，民間企業をむすぶようになっている．また海外のネットワークとも，ゲートウェイを介して接続されている．サービスの内容も電子メールだけではなく，メールとおなじ方法でやりとりされる，ニュースとよばれる広域電子掲示板もある．接続方法も通常の電話回線だけではなく，専用線やISDNなどがつかわれるようになってきた．

JUNETと技術的にはまったくおなじ方式による，小規模だが同様の機能をもったネットワークにJUICEがある．こちらはJUNETとはちがって，個人でも参加できる．JUNETとはゲートウェイがあるので，電子メールやニュースのやりとりも可能である．また海外との電子メールの交換もできる．私のところのUnixシステムはJUICEに接続されており，私の電子メールのアドレス(宛先)は，

sukeyasu@gourd.calvados.juice.or.jp

である．

電子メールをつかうととおくはなれた人とのやりとりが，手紙よりはやく，場合によっては電話よりやすくできるという利点がある．また手紙とちがってポストに投函にでかけるといった手間なしに，キーボードの操作だけで手軽におくれるし，通信の記録も自動的にコンピュータのなかにのこされる．コンピュータ同士のやりとりなので，プログラムやデータをおくりたいというときにも，大変便利である．

しかし，実際につかってみると電子メールにはそういった便利さとは別の，麻薬のような習慣性があるようだ．しょっちゅう顔をあわせていたり電話をしている相手と，用もないのに発作的に大量の電子メールのやりとりをはじめてしまったなどという体験を，かなりの人がもっているのではないだろうか．

電子メールのネットワークが以前から発達していたアメリカでは，とくに大学などで電話をしてもつかまらないし，本人をみることもめったにないが，電子メールをだすとすぐに返事がかえってくるなどということが，結構あるようだ．最近日本でも，そのような話をきくことがある．

また生活時間が一般の人とはずれているため，電子メールでは親切なのに，昼間電話してもねむそうな声ででてきて，そっけなくきられるなどということもある．なかには，目の前にいる相手をつかまえて，用があるといいながら，用件は電子メールをだすといいのこして，そのままいってしまうという，わけのわからない奴まででてきたりする．あげくのはてに，広域ネットワークで「幸福の手紙」がとびかったことさえあるらしい．

電子メールは即時性がある分，うけとるとすぐに返事をかかなければならないような，強迫観念におそわれたりもする．その結果，用もないのに無意味な手紙のやりとりをはじめてしまうのだ．

これを「山羊さん郵便症候群」と名づけてみた．さっそく電子メール仲間のAさん（といってもパソコン通信でしりあったわけではく，十数年来の友人なのだが）に，電子メールで披露したところ，即座に「私は山羊座です」というメールがかえってきた．こういうのを「山羊さん郵便」というのである．

プログラム遊工房

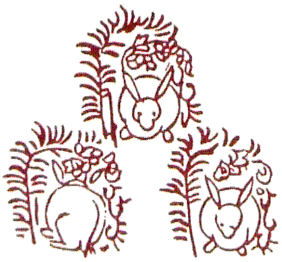
谷口典生

拾壱乃巻
「JOKE/TOKEI」プログラム

今回は簡単な「常駐プログラムもどき」を作りました．１つはVSYNC割り込みを利用したJOKEプログラム（意味のないプログラム）で，もう１つはint28hを利用した時刻表示プログラム(TOKEI）です．
ワンポイントテクニックでは，割り込みルーチンを使用する場合の注意点など，常駐プログラムについて解説します．

リスト1　joke.c

```
/*
 * joke.c              画面上の文字の色をランダムに変更する
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <dos.h>
#include <process.h>

#ifndef __TURBOC__         /* Turbo C → MS-C(Quick C) 関数名変換 */
#define enable()           _enable()
#define inportb(x)         inp(x)
#define outportb(x,y)      outp(x,y)
#define setvect(x,y)       _dos_setvect(x,y)
#define getvect(x)         _dos_getvect(x)
#endif

void (interrupt far *oldvsync)();                    /* 元のVSYNC割り込みルーチン */

void interrupt far vsync()                   /* VSYNC割り込みルーチン */
{
        unsigned int (far *attribute)[80];
        int x,y;

        attribute = (void far *)0xa2000000L;
        x = rand() % 80;                             /* 適当な座標を選ぶ */
        y = rand() % 24;
        attribute[y][x] ^= (((rand()%7)+1)<<5);  /* ランダムに色を変更 */

        oldvsync();                          /* 元のVSYNC割り込みを呼び出す */
        outportb(0x64, 0);                   /* VSYNC割り込みをリセット */
}

void set_vsync()                     /* 割り込みルーチンの設定 */
{
        outportb(2, inportb(2) | 4);         /* VSYNC割り込みをマスク */
        oldvsync = getvect(0x0a);            /* 元の割り込みベクタを保存 */
        setvect(0x0a, vsync);                /* vsync() をベクタに設定 */
        outportb(2, inportb(2) & ~4);        /* マスクのクリア */
        outportb(0x64, 0);                   /* VSYNC割り込みのリセット */
        enable();                            /* 割り込み許可フラグをセット */
}

void restore_vsync()                 /* 元の割り込みルーチンの復帰 */
{
        outportb(2, inportb(2) | 4);         /* VSYNC割り込みをマスク */
        setvect(0x0a, oldvsync);             /* 割り込みベクタを元に戻す */
}

void exec_shell()                            /* command.comの実行 */
{
        int rcode;                   /* spawn関数リターンコード(-1でエラー) */

        printf("EXITで終了します。¥n");

        rcode = spawnl(P_WAIT, getenv("COMSPEC"), NULL);
        if(rcode)
                rcode = spawnlp(P_WAIT, "command", NULL);

        if(rcode)
                printf("COMMAND.COMが実行できませんでした。");
        else
                printf("終了しました。");
}

void main()
{
        time_t seedtime;

        time(&seedtime);
        srand((unsigned int)seedtime);               /* ランダマイズ処理 */
        set_vsync();                                 /* VSYNC割り込みをセット */
        exec_shell();                                /* command.comを実行 */
        restore_vsync();                             /* VSYNC割り込みを復帰 */
}
```



プログラムの使い方

このプログラムはPC-9801シリーズ用で，Turbo C, QuickCおよびMS-Cに対応しています．リスト1，2を打ち込み，コンパイルして実行ファイルを作ってください．MS-CやQuick-Cでコンパイルするときは/Gsオプション(スタックチェックの取消)をつけてコンパイルしてください．

プログラムを実行する前に，command.comのあるパス名が環境変数COMSPECで指定されているか，またはPATHの通ったディレクトリに入っていることを確認してください．

プログラムを実行すると（JOKE, TOKEI共に引数は必要ありません），「EXITで終了します.」というメッセージが表示され，新たにcommand.comが起動します(プログラムが常駐した形になる)．JOKEの実行中は画面上の文字にランダムに色がつき，TOKEIの実行中は常に画面右上に現在時刻が大きく表示されます．どちらのプログラムもテキスト画面のアトリビュートエリアを書き換えているだけなので，さらにほかのプログラムを実行させてもとくに問題は起きないはずです．

常駐を解除して終了するには，コマンド入力の状態からEXITと打ち込んでください．「終了しました.」というメッセージが表示されてプログラムを実行する前の状態に戻ります．JOKEと

リスト2　tokei.c

```c
/*
 * tokei.c              int28hを使った時刻表示プログラム
 */

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <process.h>

#ifndef __TURBOC__      /* Turbo C → MS-C(Quick C) 関数名変換 */
#define setvect(x,y)    _dos_setvect(x,y)
#define getvect(x)      _dos_getvect(x)
#endif

#define XBASE   50                      /* 時刻表示X座標 */
#define YBASE   0                       /* 時刻表示Y座標 */

union REGS regs;                        /* intdos用 */
int i,j;                                /* put_num()用ループ変数 */
int hours, minutes, seconds;            /* 時、分、秒 */
unsigned int (far *attribute)[80];      /* VRAMアトリビュートエリアへの配列 */
int int28h_flag = 0;                    /* int28h処理中フラグ */
int numdata;                            /* 文字データ格納用変数 */
int numdatatable[11] = {                /* 0～9、:の文字データ */
        0x7b6f, 0x2493, 0x73e7, 0x79e7, 0x49ed, 0x79cf,
        0x7bcf, 0x492f, 0x7bef, 0x79ef, 0x0410 };

void (interrupt far *oldint28h)();      /* 元のint28h割り込みルーチン */

void put_num(int x, int num)            /* 一文字表示する */
{
        numdata = numdatatable[num];    /* テーブルからデータを得る */
        for(i=0; i<6; i++)
                for(j=0; j<3; j++) {
                        attribute[YBASE+i][x+j] = (numdata&1)?0xa5:0xe1;
                        numdata>>=1;
                }
}

void put_clock()                        /* 時計を表示 */
{
        put_num(XBASE, hours / 10);             /* 時(10の位) */
        put_num(XBASE+4, hours % 10);           /* 時(1の位) */
        put_num(XBASE+7, 10);                   /* ':' */
        put_num(XBASE+10, minutes / 10);        /* 分(10の位) */
        put_num(XBASE+14, minutes % 10);        /* 分(1の位) */
        put_num(XBASE+17, 10);                  /* ':' */
        put_num(XBASE+20, seconds / 10);        /* 秒(10の位) */
        put_num(XBASE+24, seconds % 10);        /* 秒(1の位) */
}

void get_time()                         /* システム時計の時刻を得る */
{
        regs.h.ah = 0x2c;                       /* 時刻の読み出し */
        intdos(&regs, &regs);
        hours = regs.h.ch & 0xff;               /* 時 */
        minutes = regs.h.cl & 0xff;             /* 分 */
        seconds = regs.h.dh & 0xff;             /* 秒 */
}

void interrupt far int28h()             /* int28hルーチン */
{
        if(int28h_flag==0) {            /* (再入ならすぐ戻る) */
                int28h_flag = 1;
                get_time();             /* 時刻を得る */
                put_clock();            /* 時計の表示 */
                oldint28h();            /* 元のint28hルーチンを呼び出す */
                int28h_flag = 0;
        }
}

void set_int28h()                       /* 割り込みベクタの設定 */
{
        oldint28h = getvect(0x28);      /* 元の割り込みベクタを保存 */
        setvect(0x28, int28h);          /* int28h() をベクタに設定 */
}

void restore_int28h()                   /* 元の割り込みルーチンの復帰 */
{
        setvect(0x28, oldint28h);       /* 割り込みベクタを元に戻す */
}

void exec_shell()                       /* command.comの実行 */
{
        int rcode;              /* spawn関数リターンコード(-1でエラー) */

        printf("EXITで終了します。¥n");

        rcode = spawnl(P_WAIT, getenv("COMSPEC"), NULL);
```

TOKEIを同時に実行することもできますが，このときは2回EXITと打ち込むことによってあとから起動したものから順に終了します．

同じプログラムを重ねて実行することも可能ですが，JOKEは最後に実行したものを終了すると画面の動きが止まってしまいます（終了時にVSYNC割り込みを禁止しているから）．画面が変化しなくてもメモリ上には前に起動したプログラムがまだ残っているので，すべて終了させるには起動した数だけEXITを打ち込まなければいけません．

プログラムの説明

リスト1(joke.c)

11-17行

MS-C(QuickC)の関数をTurbo Cの関数名に合わせるための定義

vsync()

VSYNC割り込みルーチン．乱数によって画面上の適当な座標を選び，その場所のアトリビュートを変更（ビット5～7をランダムに変更）して色を変えています．参考までにアトリビュート情報の内容を図1に示しておきます．

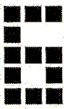
```
        if(rcode)
                rcode = spawnlp(P_WAIT, "command", NULL);

        if(rcode)
                printf("COMMAND.COMが実行できませんでした。");
        else
                printf("終了しました。");
}

void main()
{
        attribute = (void far *)0xa2000000L;

        set_int28h();                   /* int28hのベクタを設定 */
        exec_shell();                   /* command.comを実行 */
        restore_int28h();               /* int28hのベクタを復帰 */
}
```

図1　テキストVAMのアトリビュート

bit	内容
0	シークレット（0で有効）
1	ブリンク（1で有効）
2	リバース（1で有効）
3	アンダーライン（1で有効）
4	バーチカルライン（1で有効）
5	色情報（B）
6	色情報（R）
7	色情報（G）

例）　0xe1・・・・白色（通常状態）
　　　0xa5・・・・水色リバース

図2　文字キャラクタのデータ化



```
111
100
111  →  111100111101111  →  111101111001111  →  0x79cf
101      (横に並べる)         (逆順にする)
111
```

set_vsync()
restore_vsync()

VSYNC割り込みルーチンの設定，復帰をします．

exec_shell()

子プロセスとしてcommand.comを起動します．

main()

VSYNC割り込みを設定してからcommand.comを起動し，コマンド入力が可能な状態にして疑似的にプログラムが常駐した状態にします．子プロセスが終了したらVSYNC割り込みベクタを復帰してプログラムを終了します．

リスト2(tokei.c)

15-16行

時刻を画面に表示する座標の定義．この値を変更すれば時計の位置を変えることができます．

18-26行

ループ変数など，本来はローカル変数として関数内で定義するべきものですが，割り込みルーチン内でスタックの再設定をしていないので，スタックの使用をできるだけ節約するように外で定義しています．

put_num()

0から9の数字1つを3×5の大きさのキャラクタとして画面に表示します．文字キャラクタは図2のようにデータ化してnumdatatable[]に格納しています．指定された文字データの最下位ビットから順に取り出し，左上から右下に向かってアトリビュートをセットしていきます．

put_clock()

デジタル時計を表示します．時分秒を1桁ずつ取り出し，put_num()で画面に表示します．時と分の間，分と秒の間には「:」（コロン）を表示します．

get_time()

DOSのファンクションコールを利用し，システム時計の時刻を読み出します．

int28h()

int28hルーチン．処理中はint28h_flagをセットし，再入してきた場合には処理を行わずに戻ります．

set_int28h()
restore_int28h()

int28hルーチンの設定，復帰をします．

ワンポイントテクニック

常駐プログラムと割り込みルーチン

今回は，前回のプログラムで利用した割り込みルーチンを常駐プログラム(以下TSR)に応用してみました．今回のプログラムでは子プロセスとしてcommand.comを起動し，プログラムが常駐したように見せかけているだけで，本当のTSRではありません．タイトルが「常駐プログラムもどき」となっているのはそのためです．

この方法でも設定した割り込みルーチンを動かしながら，DOSのコマンドやほかのプログラムを実行できるので，TSRの雰囲気を味わうことはできると思いますが，ほかのプログラムから子プロセスとして起動させたり，BATファイルから起動するような用途には使えません．そのためには割り込みルーチンを設定したままプログラムを常駐終了し，親プロセスに処理を戻すことになります．常駐終了するだけならばTurbo Cのkeep()やMS-Cの_dos_keep() といった関数を使用すればよいのですが，この場合には常駐を解除できるようにするためにプログラムが少し複雑になるので，本当のTSRの作り方は次回以降で取り上げることにします．

また，子プロセスでcommand.comを起動する方法では，メモリを必要以上に食ってしまうという欠点もあります．TSRがあまりメモリを占有してしまうとほかのプログラムを起動するときにフリーエリアがたりなくなるといったことにもなるので，常駐する部分はできるだけ小さくすることが望ましいのです．今回は実用的なプログラムではないので，その点についてはまったく考慮されていません．JOKEやTOKEIの実行中にメモリの使用状況を見ると，子プロセスであるcommand.comの常駐部分を除いても70KB以上の領域を占有していることがわかります．JOKEやTOKEIがそんなにデータ領域を使っているわけがなく，必要以上に確保されているメモリが多いことはすぐに予想がつきます．これはプログラムが起動したときに確保されたヒープ領域のせいで，ヒープを縮小してから子プロセスを起動すれば，より多くのメモリを子プロセスに解放することができます．具体的には，Turbo Cならプログラムの中で

```
extern unsigned _heaplen = 1;
```

とすればよいし（_heaplen=0とすると最大値になってしまう），MS-C(Ver.6.0)では子プロセスを呼ぶ前に_heapmin() という関数を実行すればよいでしょう．ヒープを縮小すればJOKEやTOKEIの占有するメモリは10数KB程度になるはずです．

前回の解説でも少し触れましたが，割り込みで呼び出されるルーチンに関しては，通常のプログラムとは違った状況で動作することに注意しなければいけません．

スタックポインタの値はプログラムが起動するときにスタックエリアに設定され，使える領域は保証されています．コンパイル時に指定すれば関数が呼び出されるごとにスタックがオーバーフローしていないかをチェックする

コードを埋め込むこともできます．しかし，割り込みルーチンが呼ばれた時点ではスタックポインタの場所は不定であり，使用できるスタックの大きさもわかりません．スタックチェックのルーチンは正常に動作しないので，割り込みルーチンがほかの関数を呼ぶようなプログラムでは使ってはいけません（MS-CやQuickCではデフォルトでスタックチェックを行うため，/Gsオプションか#pragma文によって明示的にスタックチェックを行わないようコンパイラに指示する）．

Cでは関数への引数の受け渡しやローカル変数（関数内部で宣言される変数）にスタックを使うので，割り込みルーチンをCのプログラムで書くには，割り込みが発生した時点でスタックポインタを再設定し，十分なスタック領域を確保しなければいけません．スタックポインタの再設定をするにはレジスタ操作が必要で，Turbo Cでは_SS，_SPなどの予約語を使って直接レジスタの値を変更することができますが，MS-Cでは_asm文を使うことになります（今回のプログラムでは，割り込みルーチン内でローカル変数を使わず外部変数を使うようにすることでスタックを節約し，スタックポインタの設定を省いています）．

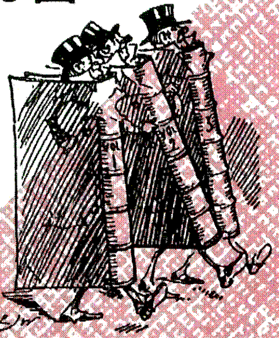
次に，再入についての問題があります．VSYNC割り込みは約60分の1秒ごとに発生するので，割り込みルーチン内での処理が長引くと，割り込みから復帰しないうちに次のVSYNC割り込みが発生し，多重に割り込みルーチンが呼ばれてしまうことになります．次の割り込みが発生しないように割り込みを止めてしまうという手もありますが，その間ほかの重要なハードウェア割り込みの処理も妨げられてしまうので，長い時間割り込みを禁止するのは望ましくありません（通信の文字落ちなどの原因にもなる）．そこで，割り込みルーチンの処理中であることを示すフラグを使い，処理が終了しないうちに次の割り込みが発生した場合はすぐ戻るようにします．

VSYNC割り込みやタイマ割り込みなどのハードウェア割り込みの処理ルーチンの中からは，基本的にDOSのファンクションコールを使うことができません．割り込みの発生がファンクションコールの処理中である可能性があるからで，BIOSコールについても同じです．どうしても使いたい場合にはDOSのファンクションコールが処理中であるかを調べてから使う方法がありますが，毎回処理中である可能性もあり，確実な方法ではありません．そのため，MS-DOSにはint28hが用意されていて，この割り込みの中からはDOSのファンクションコールの多く（ファンクション0dh以上のもの）を利用することができるようになっています．int28hはキー入力などのファンクションコールが行われたときにDOS内部から呼び出されるソフトウェア割り込みで，VSYNC割り込みのように定期的に発生する割り込みではありませんが，今回のTOKEIのように時刻を表示するといった用途には十分です．

割り込みルーチンはデバッガなどで動作を追うことがなかなかできないので，バグが発生した場合でも原因を探ることが困難です．割り込み処理のプログラムを書く場合には，基本となるプログラムの動作をよく確認し，理解してから複雑なものへ応用していくとよいでしょう．

IC千夜一夜

第6回 もうひとつの8ビットCPU
—MC6800—



虎岩 登

MC6800は，インテルの8080に遅れること約1年，1974年にモトローラから発表されたCPUです(モトローラはMPU, Micro Processing Unitと呼んでいます)．インテルとモトローラといえば，いまでもマイクロコンピュータ界を2つに分けるリーダーのように考えられていますが(最先端の32bitクラスでは，ほかのいくつかの会社に追いあげられているようですけれど)，それはこの当時からずっと続いているのです．

アメリカの半導体産業というのは，どちらかというと応用分野，つまり家電，コンピュータ，通信，軍事などの分野とは独立した会社が幅をきかせているところがあります．日本では，日立，東芝，NEC，三菱，松下，シャープなど，半導体を生産している会社は，ほとんどすべてがテレビやビデオの類から，発電所や人工衛星などの分野にいたるまで，手を広げている感があります．

しかし，アメリカではインテルやフェアチャイルド，ナショナル・セミコンダクターなど，半導体生産専門の企業が新しいICの開発をリードしています．モトローラは，その中の変わり種で，通信機器関連で有名な企業です．最近では，同社の携帯電話が日本でも話題になりました．半導体のほうでも，無線機や携帯電話に使われる狭帯域FM用変復調ICや，高周波用トランジスタ，増幅モジュールなどで優れた製品を生産しています．もちろん，デジタルICも数多く生産していました．

そのような会社が，インテルに続いて8bitマイクロコンピュータの世界へ参入してきたのです．6800とはどのようなMPUだったのでしょうか．

当時は，よく(ミニ)コンピュータ的なマイクロコンピュータと評価されていたと記憶しています．当時としては多彩なアドレッシング・モードを取り入れた命令体系や，シンプルなレジスタ構成，レジスターメモリ間の演算を基本にしていることなど，洗練された印象を受けます．ハードウェア的にも，インテルの8080と比較すれば非常にシンプルにシステムが組めるようになっています．

ハードウェアアーキテクチャ

内部にすべてを含めた設計

では，まずハードウェアの特徴から8080と比べながら見ていきましょう．パッケージは40ピンのDIPです．この点は8080と同じ，64KByteのメモリ空間と8bitのデータバスも同じです．し

かし，電源は5Vの単一電源です．また，入出力レベルも，クロック信号を除けば完全なTTLレベルで，厳密にはTTLレベルでなかった8080より使いやすくなっています（著者注：ここではインテルの8080Aのことを8080と表記しています．ちなみにAの付いていない8080では，出力端子に流せる電流が小さく，通常のTTLとは接続できませんでしたが，すぐに改良されてAのサフィックスを付けられました）．

MC6800は，メモリやI/Oをコントロールする信号を直接出力していて，ほとんど外部回路を必要としません．それに対して8080では，コントロール信号を作るための基本信号をCPUが出力していました．つまり，メモリやI/Oを接続するのに外部回路が必要でした．その代わりといっては何ですが，これからスタック領域をアクセスするとか，これから読み込むのは命令であるとか，CPU内の詳細な動作を示す信号が出力されていました．8080のような方式では，その気になればバス・タイミングを自由に設計できるのに対し，MC6800は自由度がないともいえます．8080では，回路設計者が外部に付加する回路を，MC6800では最初からMPUの内部に組み込んであるのです．さらに，これは特筆に値するのですが，モトローラはMC6800用の入出力LSIやRAMを早い時期から供給していました．これらのLSIは，最初からMC6800に接続することを前提としてタイミングが設計されていますので，ほとんど外部回路なしで直接MPUにつなげることができました．この点，インテルは8080を出してから，周辺LSIを供給するまで若干遅れがありました．少なくとも発表当時は，MC6800のほうが8080より少ないチップ構成でシステムを組み上げることができました．

バス構成では，MC6800がクロック同期型のバス，8080が非同期型のバスを採用していましたが，これはそれぞれに一長一短があるので，一概にどちらが優れているとはいえません．この同期型バスの信号タイミングは，後のMC6809など同社の8bitMPU製品についてほぼ共通になっていました．また，16bitのMPUでもMC68000やMC68010などまで，同じタイミングのバスアクセスが可能になっています．そのおかげで，周辺ICはすべて共通のものが使えました．

割り込みについては，8080はマスク可能な1レベルの割り込みしかサポートしていません（ただしこれはCPUの機能としての話であって，外部回路で増やすことができます）．また，割り込みを実際に行うには外部回路が必要でした．ところがMC6800では，マスク可能な割り込み信号と，マスクできない割り込み信号の2種類が用意されていて，しかも，その信号が加えられれば，自動的に割り込みが発生します．スタックポインタを除くすべてのレジスタがスタックに待避され，特定のメモリ領域から割り込み処理ルーチンのアドレスを読み出してジャンプします．その際に，外部回路の手助けは必要ありません．すべて，MPUの内部に含まれているのです．

このように，MC6800はできるだけ外部回路を必要としないように，内部にすべてを含めたハードウェア構成になっています．とくに，その特徴はMC6800用に用意されたファミリーLSI，つまり専用のI/Oポートやメモリを使ったときに際立って表われます．対して8080はオーダーメードのシステム用の素材を提供している雰囲気です．外部回路を必要としますが，その分自由な設計がきくのです．とにかく，

写真1　MC6800(左)，XC6820(左下)，MC6850(右)，MCM6810(右下)

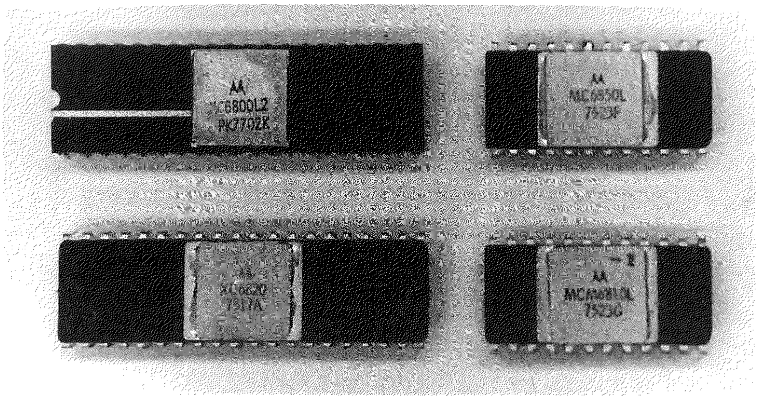
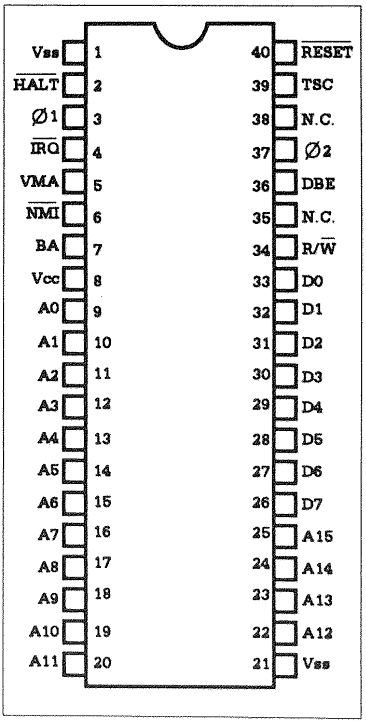


図1　MC6800のピン配置



少ないハードウェアでコンピュータ・システムを構成するのにはMC6800が向いていました.

ソフトウェアアーキテクチャ

ミニコンを手本に

では次に，ソフトウェアの面から眺めてみましょう．図2がレジスタセットです．8080と比較してもシンプルです．2つのアキュムレータ，Aレジスタと Bレジスタが演算用のレジスタで，あとはインデックスレジスタとスタックポインタ，プログラムカウンタがひとつずつあります．8080がアキュムレータとスクラッチパッドレジスタ間の演算，レジスターレジスタ間演算を基本としていたのに対し，MC6800ではレジスターメモリ間演算を基本としています．そのときの演算対象メモリのアドレス指定法，すなわちアドレッシングモードも豊富です．16bit長データでアドレス指定する拡張(絶対)アドレッシング，8bit長データで0～255番地までのアドレス(以下ダイレクトページメモリと呼びます)をアクセスする直接アドレッシング，さらにインデックスレジスタに8bit長データを加算してアドレス指定を行うインデックスアドレッシングが，メモリの番地指定に使えます．また，分岐命令では8bit長のデータで修飾するプログラムカウンタ相対アドレッシングも使えます．

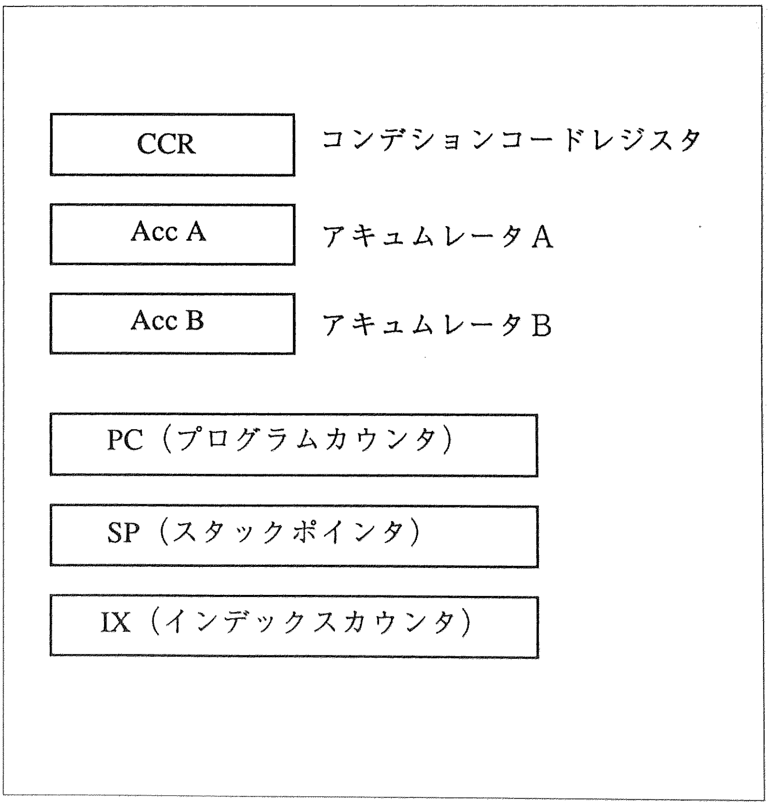
条件分岐命令の種類も8080より豊富で，数値の比較結果に対して自然な命令割り当てになっています．8080では，場合によっては，複数の条件分岐命令を組み合わせて使わなければならないところが，1命令で可能になっています．数値をレジスタにロードするだけで，フラグが変化してしまうのもMC6800の特徴です．そのほか，8086のINT命令に相当するソフトウェア割り込み命令も用意されています．

MC6800には，総じてミニコンピュータをお手本に，その機能を思いっきり削りとって小さくまとめあげたような印象があります．逆に8080では，4004からはじまったCPUを，必要に応じて8bit化して，命令の機能を上げてきた感があります．アドレッシングモードの種類も少なく，メモリの番地指定も16bit長の絶対アドレッシングと，レジスタペアを利用したポインタアドレッシングしかありません．

だからといって，MC6800が8080と比較して絶対的に使いやすかったかというと，そうともいえません．たとえば，インデックスレジスタが1本しかないので，ブロック転送を行おうとすると，インデックスレジスタの内容を何回も入れ替えなければなりません．8080なら，ポインタとして使えるレジスタペアが3本あるので，ずっと容易です．インデックスレジスタをスタックにプッシュ・ポップできないのも使いにくい点でした．したがって，どうしても1byteで直接指定できるダイレクトページメモリにインデックスレジスタを出し入れする回数が，多くなります．

また，ダイレクトページメモリは，スクラッチパッドレジスタの代わりに，一時的なデータを格納するのにも多く使われます．OSやBASICなどのシステムプログラムを併用してプログラムを組む場合は，システムプログラムが消費するダイレクトページメモリが多く，ユーザーアプリケーションが利用できる部分が少なくなってしまい，不便なこともありました．これらの欠点は，後のMC6809では徹底的に改良されています．

図2　MC6800のレジスタセット



68系の系譜

組み込み分野で健闘

このように，系統だった構成のマイクロコンピュータなのですが，8080系と比べると旗色が悪かったようです．その最大の理由は，8080と比べ約１年遅れで登場したことです．発展の著しいこの業界で，１年の遅れは致命的なものでした．性能がどうのというより，１年の間に蓄積されたソフトウェアやハードウェアの設計事例や経験のほうが，普及のためには，はるかに重要になります．

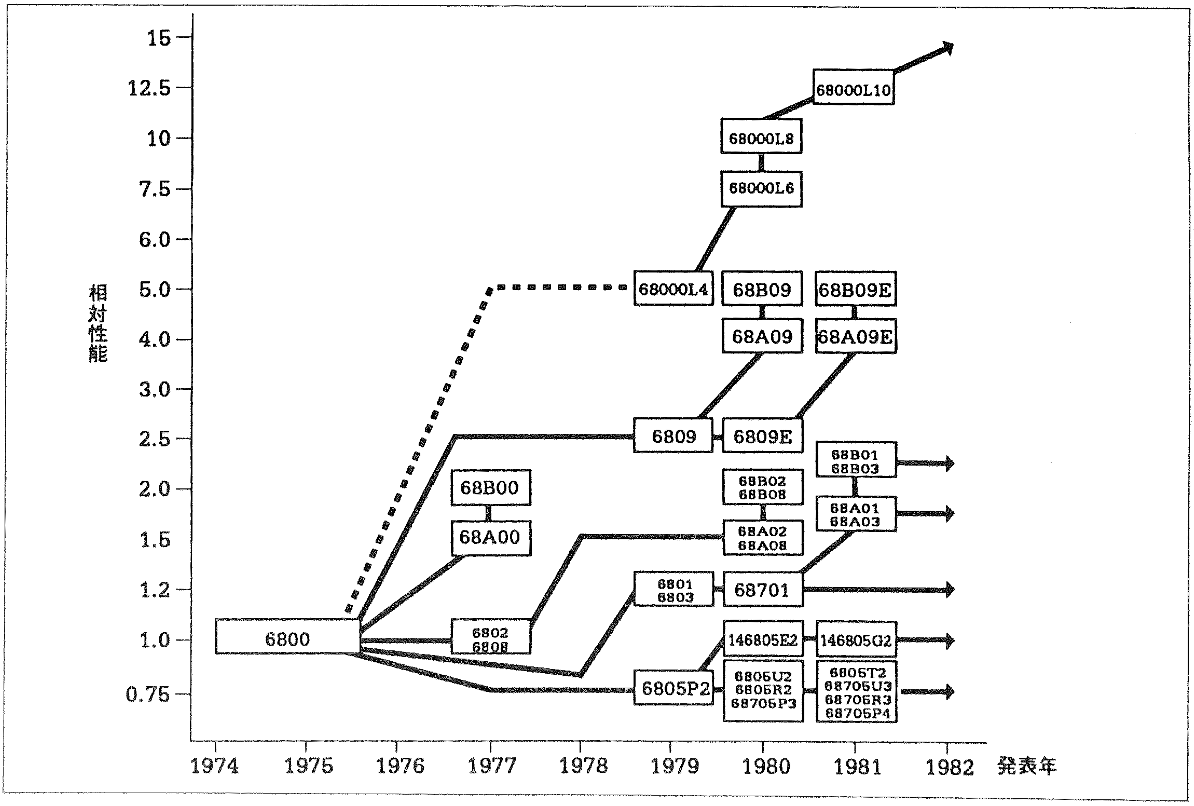
本誌1991年1月号の「パソコン博物館」にもあるように，MC6800が発表されたころには，すでに8080でBASICの動くaltair 8800が売られていたのです．ホビーからスモール・ビジネス分野までS-100busシステムが使われ，その後はAPPLE IIやPETなどの，オール・イン・ワンのパーソナル・コンピュータがCPUに6502を採用して躍進しました．SWTPC（サウス・ウエスト・プロダクツ）社製のSWTPC 6800システムというS-100bus風のコンピュータもあったのですが，こういう用途にはアメリカでは少数派でした．

ただし，機器への組み込み分野に関しては，MC6800系は健闘しているといえると思います．また，その後のモトローラ社の製品展開も，この分野を中心にフォローしていました．

MC6800にクロックジェネレータを追加したMC6808，さらにRAMを追加したMC6802という汎用MPUが1977年に発表になっています．1979年はラインアップ強化の年で，MC6800の命令をバイナリ・コンパチブルのまま強化し，RAMやI/Oポートを内蔵したMC6803と，それにROMまで加えてシングルチップ・マイクロコンピュータ化したMC6801，さらにMC6800のアーキテクチャを簡略化して命令セットを縮小したシングルチップ・マイクロコンピュータのMC6805が発表されています．

このあたりは，プリンタやモデムの内部を調べると意外なほど採用されていました．現在では，ほとんど同じアーキテクチャと命令セットで，内蔵メモリやI/Oポートの構成とかパッケージの形状が異なった組み込み用マイクロコンピュータが多種類利用できるようになっています．同年，汎用MPUとしてMC6809とMC68000も発表になっているのも忘れてはいけませんが，製品ユーザーの知らないところで使われる，組み込み用のシングルチップ・マイクロコンピュータも社会的には大切なのです．

図３



MS-DOS TOOL LABO

MS-DOSツール・ラボ

第9回 フリーメモリ残量調整プログラム REMMEM.EXE/RELMEM.EXE

加野島英渡

連載を続けてきたMS-DOSツール・ラボも今回で最終回である．連載の最後として，MS-DOSがフリーメモリを管理している機構と，その利用方法について解説する．また，フリーメモリの残量を任意に変更できるツールを作成した．

メモリ この不可思議なもの

アセンブラでツールを書き始めて戸惑うことのひとつに，メモリ管理がある．高級言語のときには，気にもとめなかったような処理が必要となってくるし，ちょっとコーディングをまちがっただけで，簡単にメモリ管理を突破して暴走する．しまいには，MS-DOSシステムの管理情報まで破壊してハングアップ……．いったいMS-DOSは，どうやってメモリを管理しているのだろうか．そしてプログラムやデータがメモリを使用するとき，どんな動作をするのだろう．今回は，MS-DOSのメモリ管理の概要とプログラムがメモリを使用するときの挙動について考えてみたい．

いかにしてプログラムは走り出すか？

たいがいのMS-DOSシステムは，システムなどを合わせて640KBのメモリ空間をもっている．その中のもっとも低位のアドレス（つまり，00000hから始まる）数KBは，CPUが使用する割り込みベクタ（00000h～003FFhの1KB）とBIOSが使用するワークエリア，そしてMS-DOSのシステムそのものとワークエリアなどに使用されており，アプリケーションプログラムは，その後方のメモリ領域を使ってロードされる．では，アプリケーションプログラムが走り出すとき，メモリはどのような状態になっているのだろうか？

通常，アプリケーションプログラムは，MS-DOSのシステムコール(INT 21h, AH=4Bh)によってロードおよび実行が開始される．この時点でのメモリ状態が，どのようになっているか

ワンラインNEWS
ウェスコムはC言語プログラム開発支援システム「CAP」を開発した．設計用のコンポーザーと既成のプログラムを解析し内容を記録するイラストレータからなる．

を理解しておかないと，アセンブラで書くプログラムの中で「子プロセスの起動」や「作業領域の獲得」のような，メモリ操作を行うのが非常に困難になってくるだろう．

MS-DOSは，プログラムの実行を開始する際に，2つのメモリブロックを作成して割り当てる．この「ブロック」というのは，メモリの1つの連続したかたまりと考えてほしい．1つのブロックには，プログラム本体，データ，スタックなどが格納される．のちほど詳しく見ていくが，このブロックはMS-DOSの2つのプログラム形式である「EXE」と「COM」によってメモリの状態がちがってくる．そしてもう1つのブロックには，そのプログラムの「環境」が格納される．「環境」というのは，AUTOEXEC.BATでおなじみのPATHや各種環境変数が入っている領域である．では，COMとEXEでどのようにメモリの割り当て方法が異なるかを見ていくことにしよう．

■COM形式のプログラム

COMプログラムでは，もっとも低位のアドレスにあるメモリブロック全体がプログラム本体のために割り当てられる．つまり，COMプログラムを実行したときには，MS-DOSが管理している残りメモリをすべて，そのプログラムに割り当てられて立ちあがってくるのである．

■EXE形式のプログラム

これに対してEXEプログラムでは，起動されたときのメモリ最大割り当て量をリンク時に決定できる．この情報は，EXEヘッダと呼ばれる情報に組み込まれていて，起動時にMS-DOSによって読みだされる．MS-DOSは，起動するプログラムが指定したサイズのメモリを（可能ならば）割り当てるのである．残りメモリ量が指定の最大割り当て量より少ない場合は，できる限りのメモリが用意される．ところが，EXEプログラムのメモリ最大割り当て量のデフォルト値は1MBである．これは，なんの指定もしない場合は，起動時の残りメモリをすべて獲得するということを意味している．結局，EXEプログラムもおおかたの場合に，COMプログラムと同様に，残りメモリをすべて割り当てられて起動しているのである．

では，EXE形式のプログラムをもう少し詳しくみてみよう．まず図2は，メモリにロードされる前のディスク上に存在するEXEプログラムのファイル形式である．手元にあるEXE拡張子のファイルをDUMPコマンドなどで見て確認されたい．

図2のヘッダのなかに「ロードされたプログラム本体に追加されるメモリの最大パラグラフサイズ」という項目があるが，この項目がさきほど触れた起動時のメモリ最大割り当て量である．たいていのプログラムは，この値がFFFFh（実質的には「残りメモリすべて」を意味する）になっているはずである．

「パラグラフサイズ」という用語が出てきたが，パラグラフの単位は16バイトである．MS-DOSのメモリ管理は，この16バイトが最少単位となるので，1バイト使うためにも16バイトを消費する．つまり，実際に必要とするメモリ容量を16で割って整数に切り上げた大きさがパラグラフサイズということになる．

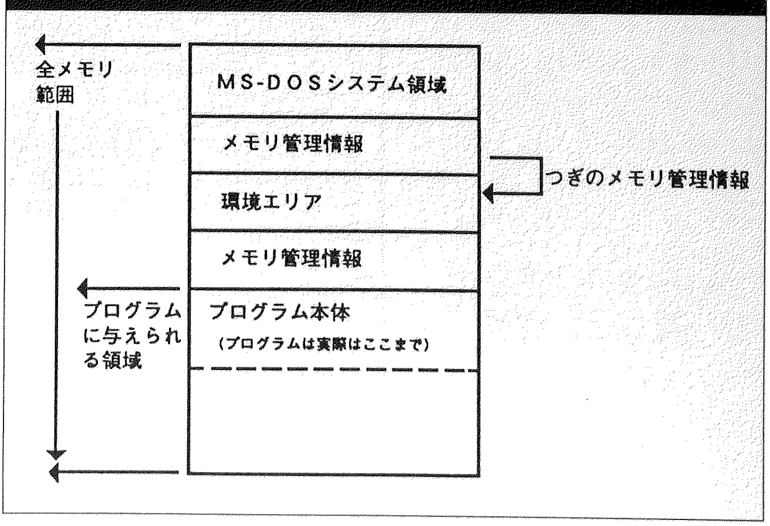
図2のヘッダは，MS-DOSがプログラムのロードの際に使用する情報を格納しているものであり，実行を開始するときにはメモリ上には残らない．MS-DOSは，プログラム実行時に「PSP(Program Segment Prefix)」と呼ばれる情報を，プログラム本体の直前に付加する．

お釣りの返しかたのマナー

先にも書いたように，MS-DOSのプログラムは起動時に自分自身で残りメモリをすべて使い切っている状態になっており，子プログラムを起動することも作業領域を獲得することもできない．

では，どうやったらプログラムの中

図1　プログラムロード時のメモリの状態



でメモリを獲得できるのか？

結局，あるプログラムが起動したときには，ほとんどの場合に残りメモリを使い切っていることがわかった．では，よく聞く話のように「プログラムの中でメモリ獲得してワークなどに使用する」ためにはどうしたらいいのだろうか?

いらないメモリは返せばいい

ということである．こう書いてしまうと身もふたもないので，もうちょっと詳しく説明することにしよう．これは，7月号で説明をはしょって，作業をリンカに押し付けてしまった事項である．MS-DOSのメモリ管理サービスは，すべて「メモリブロック」単位に行われている．そして，メモリブロックの操作が直接できるシステムコールは，図4の3つである．

ここで，メモリブロックの解放というサービスがあることが判明した．では，これを使えばいいのだろうか．いらないブロックは返せばいいといっても，先に書いたように起動されたプログラムは，それ自身のメモリブロック1つで，残りメモリ全部を使い切っているので，これを解放してしまうということは，自分自身すべてを解放してしまうことになってしまう．これでは自分がお化けになってしまうので，この場合の正解は「サイズ変更」を使用するである．

いらないメモリの返しかた実例

「メモリブロックのサイズ変更」というシステムコールをもう少し詳しく見てみよう．パラメータおよび戻り値は図5の通りである．

図2　EXEファイルの構造

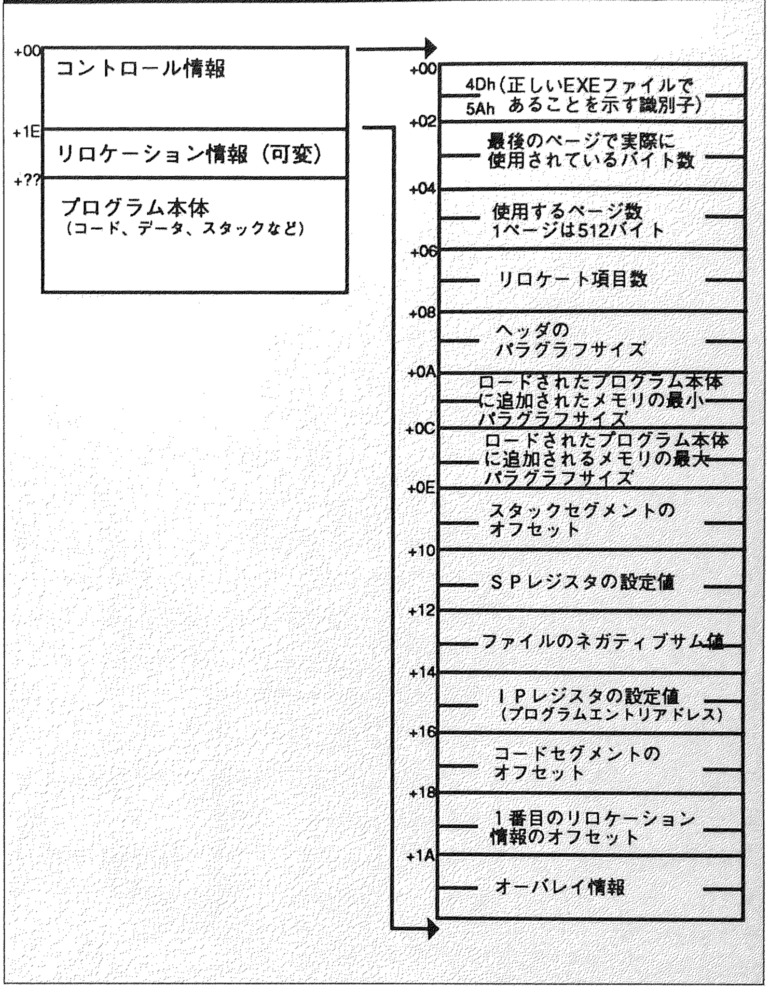


図3　EXEファイルのロード直後の状態

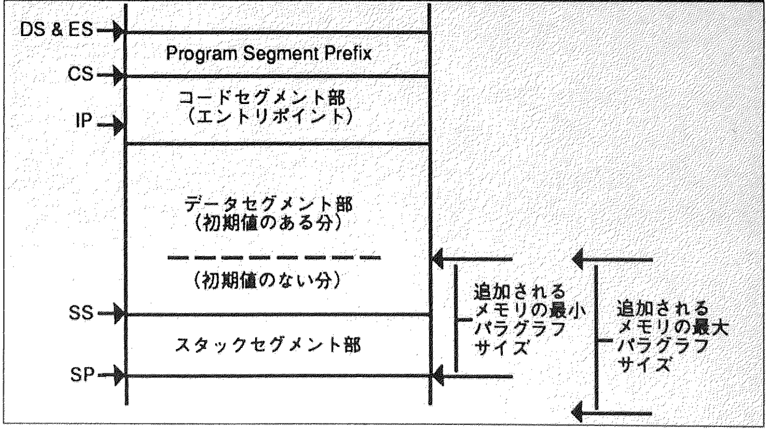


図4

ＡＨ＝４８ｈ　メモリブロックの割り当て
ＡＨ＝４９ｈ　メモリブロックの解放
ＡＨ＝４Ａｈ　メモリブロックのサイズ変更

ワンラインNEWS
ビー・ユー・ジーは「Macintosh」で構成したLANをISDNで接続する装置「ROUTE ONE」を発売した．NTTのINSネット64を使い，最高128Kbpsでデータを転送できる．

戻り値にエラーが返ってくるのは，変更によっていまよりサイズを大きくしようとしたときである．希望のサイズにすることができなかった場合は，変更できる最大のパラグラフサイズが返ってくる．これを利用して，現在獲得できるメモリ容量の上限を知ることができる．

プログラムの起動時に余分なメモリを返すには，「自分自身が必要とするサイズ」を変更後のパラグラフサイズとしてサイズ変更を行えばよい．では，どうしたら自分自身が必要とするサイズを得ることができるだろうか．EXE形式とCOM形式で算出方法が異なるので，それぞれについて考えてみる．

■EXE形式の場合

前に書いたように，EXE形式のプログラムはコーディングによってロードされるセグメントの順番が変わってくる．図6では，スタックセグメントが最終セグメントとなっている（先頭は常に環境領域のセグメントであるが，最終セグメントは必ずしもスタックセグメントであるとは限らないので注意してほしい）．この場合を考えれば，実際に必要となるパラグラフサイズは，

最終セグメント－環境セグメント＋最終セグメントのパラグラフ長

ということになる．

■COM形式の場合

COM形式の場合はもうすこし単純である．プログラム先頭が環境セグメントであるのは，EXE形式と同じであるが，COM形式のプログラムは常にスタックポインタが先頭から64KB目を指している．機械的に64KBへとメモリサイズ変更を行ってもよいが，スタックはそんなにいらないという場合には，自分でスタックポインタをかさ上げして，そこまでを必要メモリとして残すこともできる．

ここで，普段CやPascalなどのコンパイラを使っている読者は，「そんなことははじめて聞いたが，子プログラムの起動もちゃんとやっていた」と疑問に思われるかもしれない．たいていの高級言語では，スタートアップルーチンなどと呼ばれる前処理で，余分なメモリの返還を自動的にやってくれている．しかし，アセンブラではそうはいかないのだ．

これがメモリ管理だ

いままで書いてきたように，MS-DOSのメモリは，すべて「メモリブロック」という16バイトを最小とする単

図5

＜＜パラメータ＞＞
　ＡＨ＝４Ａｈ
　ＢＸ＝メモリブロックの変更後のパラグラフサイズ
　ＥＳ＝サイズ変更を行うメモリブロックのセグメント

＜＜戻り値＞＞
　キャリーフラグ＝正常終了なら０、エラーなら１
　ＢＸ＝（エラーの時）変更できる最大パラグラフサイズ

図6

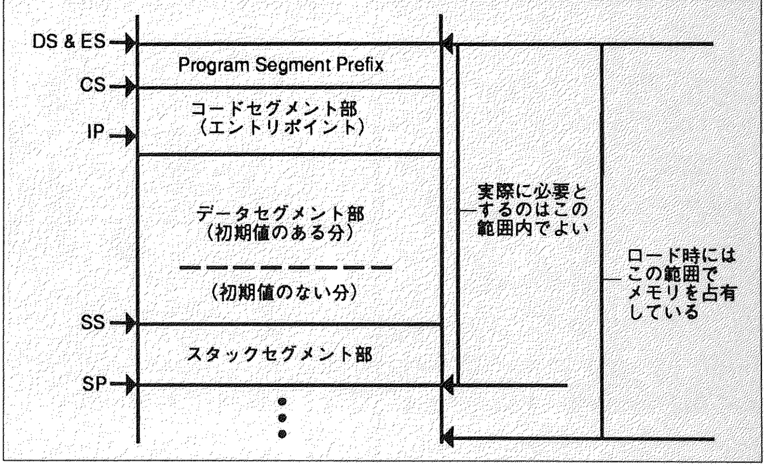
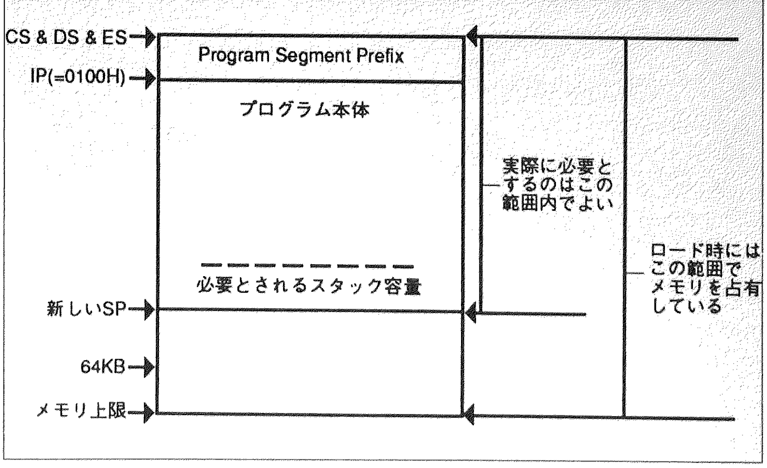


図7



位に分けられて管理される．そして，このメモリブロックには，おのおのメモリ・コントロール・ブロック（以下MCBと略記する）という管理情報が付加されている．いま一度，冒頭のプログラムロードのときの図を思い出していただきたい．プログラムに付随する環境領域，そしてプログラム本体，それぞれにメモリ管理情報が付けられている．これがMCBである．

MCBは，16バイトのテーブル構造になっている．図を見てわかるように，実際に使用しているのは先頭の5バイトほどであるが，なぜ16バイトになっているのかは容易に察しがつくだろう．MCBは簡単な構造をしている．含まれる情報は

- このMCBが最終であるかどうか
- このMCBを使用しているプログラムまたはデータのセグメント
- このMCBが管理するメモリブロックのサイズ

この3つだけである．簡単であるが，MCBをたどっていってこの3つの情報を総合することにより，MS-DOSが管理しているすべての領域の状態を知ることができる．MCBは，通常はプログラムからは見えない位置にある．この情報を破壊すると，MS-DOSのメモリ管理が破綻してシステムが停止してしまうので，参照や操作には十分な注意が必要だからである．

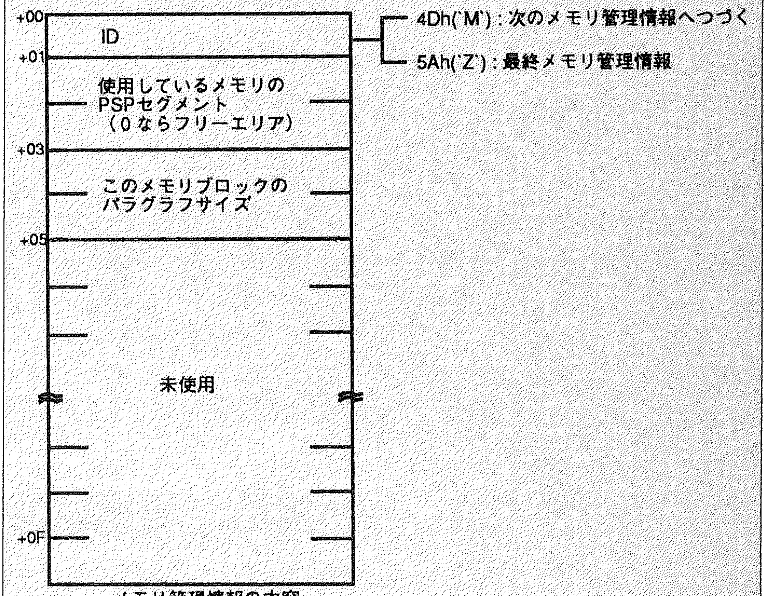
メモリコントロールブロックの歩きかた

では，このMCBを使って，いかにして残りメモリなどの情報を得るかを見ていくことにしよう．まず得るべきなのは，先頭のMCBのアドレスである．これがわかれば，あとはそのMCB自身の中に次のMCBのありかを示すデータが入っているので，順にたどっていけばよい．先頭のMCBのアドレスは図9のようにして取得する．

MCBのオフセットは，常に0であるので，これで，最初のMCBをアクセスすることが可能になる．MCBが見つかったら，まずチェックするのは，先頭の1バイトに'M'または'Z'というデータが格納されているかどうかである．このいずれかが入っていない場合は，MCBが破壊されているとみなして処理を継続すべきではない．また，'Z'が入っていた場合は，このMCBがメモリ中の最終MCBである．メモリブロックの大きさを知るためには，MCBの4バイト目，5バイト目に入っているパラグラフサイズを参照する．そして，このメモリブロックが，実際にいずれかのプログラムで使用されているかどうかはMCBの2バイト目，3バイト目の所有PSPセグメントを見るとわかる．このワードが0であれば，このメモリブロックは未使用である．

1つのメモリブロックの処理が終ったら，次のMCBを探して同じ処理を行う．MCBは16バイト（1パラグラフ）であるので，次のMCBは，

図8



メモリ管理情報の内容

＊このメモリ管理情報自身のセグメント値+1にパラグラフサイズを足すと、次のメモリ管理情報のセグメントが算出できる。

図9

```
<<パラメータ>>
 AH＝52h
 INT  21h

<<戻り値>>
 ES：BX－2   最初のMCBのセグメント
```


ワンラインNEWS
トーキンは低温，高温相互間のデータ互換性を向上させたFD「ニューSQシリーズ」を発売した．新技術を導入し，高温領域での耐久性を20%以上向上させた．

1（MCB自身）＋メモリブロックのパラグラフサイズ＋現在のセグメント

で求めることができる．

あとで実際のプログラムによってMCBをたどってみるが，概要をつかむためにリスト１のコーディングを参照されたい．

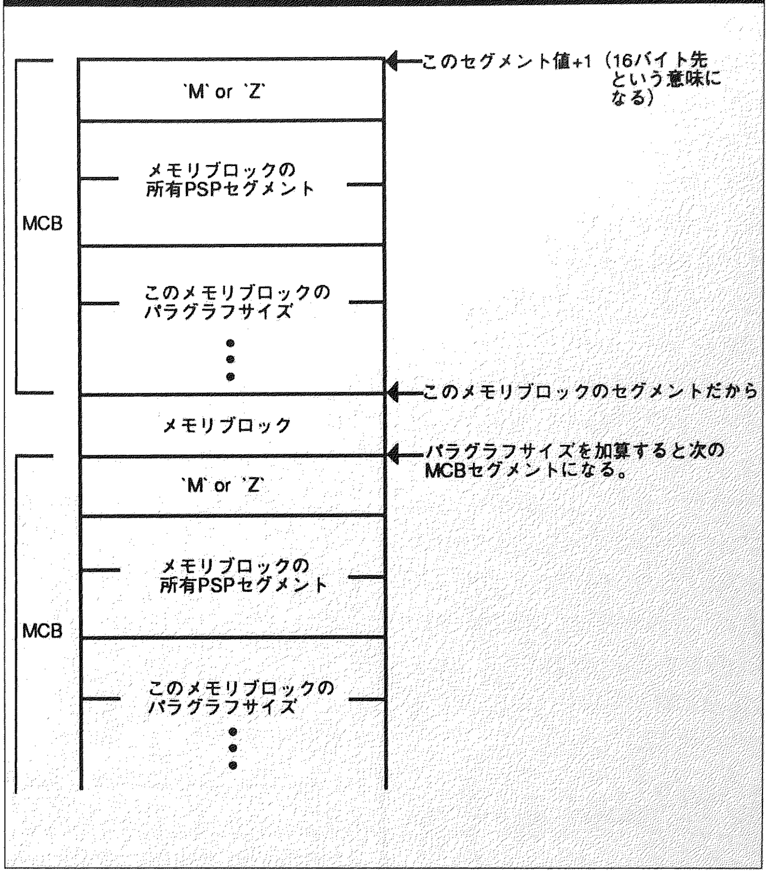
今月のプログラム

普段使うようなものではないが，もっているとなにかのときに役にたつかもしれないというツールを作成してみる．このツールは

「指定した容量分だけフリーメモリを残す」

という機能をもっている．こんなもの何に使うのか？　と疑問に思われる方もいるかもしれないが，たとえば，ちょっとしたプログラムを作成してテス

図10



COLUMN　恐怖の386専用命令

仮想86モードもさることながら，386CPUにはリアルモードでも恩恵のある命令が拡張されている．FS，GSという２つのセグメントレジスタもありがたいし，ビットフィールド操作命令はグラフィックデータの操作などに威力を発揮しそうである．なかでも筆者がひそかに気に入っているのは，条件ジャンプがNEARジャンプ可能になったことである．従来だと条件ジャンプはSHORTに限られていたため，ちょっと条件ジャンプの下の処理が込み入ってくると

```
        JZ  _NEXT
        JMP _END
_NEXT:
```

などというコーディングをしなければならなかった（こうならないようにすっきりしたコーディングをするのがウデだ，という指摘はおいておくことにする）．386CPUだと条件ジャンプにNEARが使えるので，こういったコーディングはかなり減るはずである．

筆者の使用機もいちおう386機のはしくれであるので，この恩恵に浴する権利がある．ただ，386機以外でこのコードを動かすと，当然のことながら未定義命令となるので，CPUチェックをしてから実行するなりの対策が必要である．

１度これでひどい目にあったことがある．マイクロソフトのマクロアセンブラでは386固有の命令をアセンブルするために「.386」の擬似命令があるのだが，この指定をしたときの条件ジャンプのデフォルトが，なんとNEARジャンプなのである．286でも動作させるためには「JZ SHORT ・・・」と明示しなければならない．うっかりと冒頭で「.386」指定をしてCPUチェックをしたものだから，286機にもっていくとみごとに例外割り込みとなる．たしかに，マニュアルには「386CPUで実行するプログラムにのみ使用するべきである」と書かれてはいるが……．ソースを見てもしばらく気がつかず往生したものだ．一見して386専用とわかる拡張命令なら事前に注意するのだが，こういう落とし穴もあるので注意されたい．

トランする場合，残りメモリが非常に少ないときに，異常な動作をするという障害が発生したとする．これを再現して追跡調査するためには，任意のフリーエリア容量がほしい．いろいろと常駐ソフトやデバイスドライバを組み込んで望みのフリーエリアにすることもできるが，手間がかかるうえ，組み込んだプログラムによる副作用も心配である．

このツールは，指定容量のフリーエリアを作るために，相応のメモリを獲得して死蔵する以外には副作用を起こさない（はずである）．ほかにも「このプログラムは最低何KBの空きがあれば動くのか？」を知るのにも使えると思う．このために，COMMAND.COMをいくつも起動していたという，昔の筆者のような方には有効に使ってもらえるだろう．

動作説明

■メイン

パラメータを解析して，エラーであればメッセージを表示して終了する(get_param)．

自分自身の余分なメモリを解放する(resize_mem)．

MCBチェインをたどって，メモリの総量および未使用メモリの合計を計算する（mcb_check)．

この時点で，MCBチェインが正しくたどれなかったらメッセージを表示して終了する．

「総メモリ」「使用済みメモリ」の表示を行う．

残りメモリ（総メモリ－使用済みメモリ）>指定パラメータならば，指定パラメータを表示して，残りメモリと指定パラメータの差だけメモリ獲得する．

獲得したメモリを解放しないように常駐終了する．

エラーで終了する場合は，非常駐終了とする．

■get_param

set_paramを呼び出す．パラメータは以下の通り．

DS:SI

起動パラメータ格納位置（環境領域のセグメントおよびオフセット81h)

ES:DI

解析済みパラメータ格納アドレス

CX

パラメータ要素数（当プログラムの場合は1）

リスト1

```
リスト1

        mov     ah,52h
        int     21h                     ; システムコールによってMCBｾｸﾞﾒﾝﾄを取得
        mov     ax,es:[bx-2]            ; 先頭のＭＣＢセグメント
        xor     si,si                   ; ＭＣＢのオフセットポインタに使う
        mov     bx,0                    ; このレジスタにメモリ合計を累計
        mov     cx,0                    ; このレジスタに未使用合計を累計

loop:
        mov     ds,ax
        cmp     byte ptr [si],'Z'
        jz      ok
        cmp     byte ptr [si],'M'       ; 'Z'か'M'でないとエラーとする
        jnz     ng
ok:
        add     bx,[si+3]               ; メモリ合計への加算
        cmp     word ptr [si+1],0       ; ＰＳＰセグメントが０以外なら使用中
        jnz     next
        add     cx,[si+3]               ; ０の場合は未使用として加算する
next:
        cmp     byte ptr [si],'Z'
        jz      end                     ; 'Z'だったらこれで処理終了
        inc     ax                      ; つぎのＭＣＢアドレスの計算
        add     ax,[si+3]               ;
        jmp     loop
                .
                .
                .
```

リスト2　dosfunc.h

```
;;;;;;;;;;;;;;;;;;;;;
;       dosfunc.h     ;
;;;;;;;;;;;;;;;;;;;;;
;
H_STDOUT        equ 1           ; 標準出力ハンドル
H_ERROR         equ 2           ; 標準エラー出力ハンドル
;
NORMAL_END      equ 0           ; プロセス正常終了コード
;
AT_A            equ 00100000b   ; アーカイブ属性
AT_D            equ 00010000b   ; サブディレクトリ
AT_V            equ 00001000b   ; ボリュームラベル
AT_S            equ 00000100b   ; システムファイル
AT_H            equ 00000010b   ; 隠しファイル
AT_R            equ 00000001b   ; リードオンリーファイル
;
ATTR_ALL        equ AT_A + AT_D + AT_V + AT_S + AT_H + AT_R
;
; === ＩＮＴ－２１Ｈ機能番号 ===
;
DOS_SETDTA      equ 1ah         ; ＤＴＡアドレス設定
DOS_SETVECT     equ 25h         ; 割り込みベクタ値の設定
DOS_GETVECT     equ 35h         ; 割り込みベクタ値の取得
DOS_HWRITE      equ 40h         ; ファイル／デバイスへの出力
DOS_TERMINATE   equ 4ch         ; プロセス終了
DOS_STAYTERM    equ     31h                     ; プロセス常駐終了
DOS_FINDFIRST   equ 4eh         ; 最初のファイルの検索
DOS_FINDNEXT    equ 4fh         ; つぎのファイルの検索
DOS_MALLOC      equ 48h         ; メモリ獲得
DOS_RESIZEMEM   equ 4ah         ; メモリブロックサイズ変更
DOS_MRELEASE    equ 49h         ; メモリ解放
DOS_GETLIST     equ 52h         ; MCB セグメント取得等
;
DOS_PRINTCHR    equ 02h         ; 一文字表示
DOS_PRINTSTR    equ 09h         ; 文字列表示
DOS_GETDATE     equ 2ah         ; 日付の獲得
DOS_GETTIME     equ 2ch         ; 時刻の獲得
DOS_CREATEFILE  equ 3ch         ; ファイルの新規作成
DOS_OPENFILE    equ 3dh         ; ファイルのオープン
DOS_CLOSEFILE   equ 3eh         ; ファイルのクローズ
DOS_READFILE    equ 3fh         ; ファイルの読み込み
DOS_WRITEFILE   equ 40h         ; ファイルの書き込み
DOS_SETFILEPTR  equ 42h         ; ファイルポインタの移動
```


ワンラインNEWS
アップル・コンピュータは今秋からソ連や東欧へ「Macintosh」を輸出する．ソ連や東欧でも研究者の間で人気が高かったが，本格的な販売活動は行っていなかった．

CXに0が返ってきた場合はエラーとする．

解析済みパラメータを10進のバイナリに変換する．数字以外のキャラクタが出てきたところで打ち切りとするが，このときに1桁も変換していなかったらエラーとなる．

入力されたパラメータはKB単位であるので，これをパラグラフ単位に直す．

入力値＊1024／16で変換できる（結果的に256を掛ければよい）．

■resize_mem

常駐終了するときのために，自分自身の環境エリアを解放する．

環境セグメントとスタックセグメントの差を計算する．

それに，スタックの大きさ分のパラグラフサイズを加算する．

この結果が自分自身に必要な領域であるので，システムコールを使用してこのサイズにサイズ変更を行う．

エラー情報は，DOSから返ってくるものをそのままメインに戻す．

■mcb_check

システムコールを使用して，最初のMCBのセグメントを取得する．

ここから以下の処理を繰り返す．

MCBの先頭が'M'か'Z'以外ならエラー終了する（MCBが不正である）

MCBの先頭＋3バイト目のパラグラフサイズをメモリ総量に加算する．

MCBの先頭＋1バイト目のPSPセグメントが0になっていたら未使用メモリにも加算する．

（このMCBセグメント）＋（MCBの先頭＋3バイト目）＋1を計算して次のMCBのアドレスを求める．

MCBの先頭が'Z'でなければ，以上の処理を繰り返す．

■アセンブルとリンクの方法

今回は，連載の2回目（1991年5月号）でご紹介したコマンドラインオプション処理ルーチン「SET_PARAM」を使用しているので，このルーチンを含むモジュール「SETPARAM.ASM」とのリンクが必要になる．

```
A> MASM REMMEM;
A> MASM SETPARAM;
A> LINK REMMEM+
SETPARAM,REMMEM;
```

注：”DOSFUNC.H” は先月号までのものに追加を行っている．

■使い方

```
A> REMMEM nnnn <ret>
```

コマンドラインパラメータで，「実行後に残したいメインメモリ容量」を指定する．単位は「KB」である．指定さ

リスト3 remmem.asm

```
PAGE            60,128

.xlist
include         dosfunc.h
.list

                extrn   set_param:near

_TEXT           SEGMENT WORD PUBLIC 'CODE'
_TEXT           ENDS

_DATA           SEGMENT WORD PUBLIC 'DATA'
_DATA           ENDS

_STACK          SEGMENT WORD STACK
_STACK          ENDS

DGROUP          GROUP   _DATA,_STACK
                ASSUME  DS:DGROUP, SS:DGROUP

;
; ========== 定数定義 ========================
;
PSP_PARAM   equu    80h
ARGMAX      equ     1
ARGBUFSIZE  equ     10
STACK_SIZE  equ     256
;
; ==========================================
;
_TEXT           SEGMENT
        ASSUME CS:_TEXT

;--------------- REMMEMが常駐していることを確認するための識別文字列

        db      'REMMEM'

;--------------- パラメータをチェックしてバイナリに変換する
; CY --> 1なら異常
; AX --> 変換されたパラメータ（パラグラフ単位）
get_param proc
        push    es
        mov     ax,DGROUP               ; === ｺﾏﾝﾄﾞﾗｲﾝﾊﾟﾗﾒｰﾀ取得 ===
        mov     es,ax                   ;     環境領域のセグメント
        mov     si,PSP_PARAM+1          ;     環境領域のオフセット
        mov     di,offset param         ;     ﾊﾟﾗﾒｰﾀをｾｯﾄする領域
        mov     cx,ARGMAX               ;     ﾊﾟﾗﾒｰﾀの要素数
        call    set_param
        pop     es
        mov     ds,ax
        jcxz    param_error             ; 一桁も変換できなければエラーとする

        mov     si,offset param+2
        xor     bx,bx                   ; 数値の積算用
        xor     cx,cx                   ; 桁数のカウンタ
trans_loop:
        mov     al,[si]
        cmp     al,'0'                  ; '0'～'9'以外が出てきたら終わり
        jb      trans_end
        cmp     al,'9'
        ja      trans_end
        sub     al,'0'                  ; 数値に変換する
        add     bx,bx                   ; 積算した数値を10倍して足し込む
        mov     dx,bx
        add     bx,bx
        add     bx,bx
        add     bx,dx
        xor     ah,ah
        add     bx,ax
        inc     si                      ; ポインタ更新
        inc     cx                      ; 桁数更新
        jmp     trans_loop
trans_end:
        jcxz    param_error
        mov     ax,bx                   ; パラグラフ単位に変換する
        mov     cx,40h
        xor     dx,dx
        mul     cx
        clc
        ret
param_error:
        stc
        ret
get_param       endp

;--------------- メモリサイズの変更
resize_mem      proc
        push    es
        mov     ax,es:[002ch]           ; 自分自身の環境エリアを解放する
        mov     es,ax
        mov     ah,DOS_MRELEASE
        int     21h
        pop     es

        mov     ax,es                   ; 環境ｾｸﾞﾒﾝﾄが先頭
        mov     bx,ss                   ; ｽﾀｯｸｾｸﾞﾒﾝﾄが末尾
        sub     bx,ax                   ; その差をとって・・

        mov     ax,STACK_SIZE / 16      ; スタック長を加え(パラグラフ長)
        add     bx,ax
```

れた値が無効（数値でない，指定がないなど）である場合とメインメモリがすでに指定分より少ない場合にはエラーメッセージを表示して終了する．

今月のお勧め

今回作成した「指定したメモリ残量を設定するツール」には，いったん設定した残り容量を元に戻す機能がない．「リセットをすると元に戻る」ということにしようと思ったが，それではあまりにも申し訳ない．そこで，今回作成したツールによって確保したメモリブロックを検索して解放するツール「RELMEM」を付録とする．このツールは，REMMEMの常駐位置をMCBから検索して，該当ブロックを解放する機能をもっている．REMMEMが2重3重に常駐していても，そのすべてを検索して解放するので，メモリの状態は元にもどることになる．常駐ソフトの解放の方法の1つとして見ていただきたい．

```
        inc     bx
        mov     my_size,bx              ; サイズ変更を保存しておいて
        mov     ah,DOS_RESIZEMEM        ; サイズ変更を実行する
        int     21h
        ret
resize_mem endp

;--------------- ＭＣＢチェインからメモリ容量を計算する
; CY --> 1なら異常
; BX --> 総メモリパラグラフ数
; CX --> 使用中メモリパラグラフ数
;
mcb_check       proc
        mov     ah,DOS_GETLIST
        int     21h
        mov     ax,es:[bx-2]            ; MCBセグメント取得
        xor     si,si                   ; アドレスポインタに使う
        xor     bx,bx                   ; メモリの総量をカウントする
        xor     cx,cx                   ; メモリの空きをカウントする
mcb_loop:
        mov     ds,ax
        cmp     byte ptr [si],'M'       ; 「M」なら続きがある
        jz      mcb_ok
        cmp     byte ptr [si],'Z'       ; 「Z」なら最終ＭＣＢ
        jnz     error_end               ; それ以外なら異常
mcb_ok:
        add     bx,[si+3]               ; ＭＣＢ中のサイズ積算
        cmp     word ptr [si+1],0       ; ＰＳＰセグメントをチェックして
        jz      next                    ; ０なら未使用ブロック
        add     cx,[si+3]               ; 使用中メモリのカウント
next:
        inc     ax                      ; 次のＭＣＢのアドレス計算
        add     ax,[si+3]
        cmp     byte ptr [si],'Z'       ; 「Z」を処理したら終了
        jnz     mcb_loop
        ret

error_end:
        stc
        ret
mcb_check       endp

;--------------- パラグラフをＫＢ単位として見出し付きで表示する
; AX <=== 表示する値（パラグラフ単位）
; DX <=== 表示する見出しのオフセット
disp_mem_value  proc
        push    ax
        mov     ah,DOS_PRINTSTR         ; 見出しを表示する
        int     21h
        pop     ax

        push    bx
        push    cx                      ; ＫＢ単位に直す
        mov     cl,6                    ; nnnn / 64
        shr     ax,cl
        inc     ax

        mov     cx,2                    ; 最大でも３桁で収まる
        mov     si,offset num_table
word_loop:
        xor     dx,dx
        mov     bx,[si]
        div     bx                      ; 各々の桁で割って商を表示して
        push    dx
        add     al,'0'
        mov     dl,al
        mov     ah,DOS_PRINTCHR
        int     21h
        pop     dx
        mov     ax,dx                   ; その余りで次の処理へ
        inc     si
        inc     si
        loop    word_loop
        add     dl,'0'
        mov     ah,DOS_PRINTCHR
        int     21h
        mov     dx,offset mes_end       ; 終端の文字列を表示
        mov     ah,DOS_PRINTSTR
        int     21h
        pop     cx
        pop     bx
        ret
disp_mem_value  endp

;---------------------------------------------------------------
main    proc    near
        call    get_param               ; パラメータをバイナリに変換
        jc      error_para              ; 正常に変換できなければエラー
        mov     para_value,ax

        push    ds
        push    es
        call    resize_mem              ; 自分の余分なメモリを解放して
        call    mcb_check               ; ＭＣＢからメモリの総量を計算し
        jc      error_mcb
        pop     es
        pop     ds

        mov     ax,bx
        mov     dx,offset all_mem_mes   ; 「総メモリ」
        call    disp_mem_value

        mov     ax,cx
        mov     dx,offset use_mem_mes   ; 「使用済みメモリ」
        call    disp_mem_value

        sub     bx,cx                   ; 現在の残りメモリチェック
        mov     ax,para_value
        sub     bx,ax                   ; 残りメモリ - 指定残りサイズ
        jbe     error_mem               ; そんなに残っていなければエラー

        dec     ax
        mov     dx,offset req_mem_mes   ; パラメータで指定された残り容量
        call    disp_mem_value

        dec     bx
        mov     ah,DOS_MALLOC           ; 差分をメモリ獲得して終了
        int     21h
        jc      error_mem
        mov     [block_seg],ax

;--------------- 正常終了（メモリを解放されないため常駐する） ------
        mov     ah,DOS_STAYTERM
        mov     dx,my_size
        int     21h
;--------------- エラーリターン（常駐しない） --------------------
error_para:
        mov     dx,offset error_para_mes ; パラメータ指定エラー
        jmp     short nop_return
error_mcb:
        mov     dx,offset error_mcb_mes ; ＭＣＢが不正になっている
        jmp     short nop_return
error_mem:
        mov     dx,offset error_mem_mes ; メモリ残量が足りない
nop_return:
        mov     ah,09h                  ; エラーメッセージを表示して
        int     21h

        mov     ax,4C00h                ; 非常駐終了
        int     21h
main    endp
; ======================================================================

_TEXT   ENDS

_DATA   SEGMENT WORD PUBLIC 'DATA'

param           dw  ARGMAX      dup (?)
                db  ARGBUFSIZE  dup (?)

block_seg       dw      0
para_value      dw      ?
my_size         dw      ?
all_mem_mes     db      0Dh,0Ah
                db      'ユーザメモリ・・・・・・・・$'
use_mem_mes     db      '使用済メモリ・・・・・・・$'
lest_mem_mes    db      '残りメモリ・・・・・・・・・$'
req_mem_mes     db      '設定後残りメモリ・・・$'
mes_end         db      ' KB',0Dh,0Ah,'$'

num_table       dw      100
                dw      10

error_para_mes db   0dh,0ah,'パラメータは１０進（単位ＫＢ）で指定します。'
               db   0dh,0ah,'$'
error_mcb_mes  db   0dh,0ah,'メモリコントロールブロックが異常です。'
               db   0dh,0ah,'$'
error_mem_mes  db   0dh,0ah,'空きメモリが既に指定以下になっています。'
               db   0dh,0ah,'$'

_DATA   ENDS

_STACK  SEGMENT STACK
                db      STACK_SIZE  dup (?)
_STACK  ENDS

        END     main
```


ワンラインNEWS
ソニーは同社の「NEWSシリーズ」用の新型OS「NEWS-OS 4.1」を発表した．CISC型用とRISC型用の２種類に分け，従来の4.0に比べ最大30倍に描画を高速化した．

今月の反省

今月でこの連載も最後となる．

アセンブラでDOSのツールを書いて，読者にご説明申し上げろとの編集部からのお達し．はいはいとお引き受けしたものの，昨今はこんなツールはないものかと，ちょっとネットワークに入れば必ずあるというありがたい時代である．編集部期待の「なにか役に立つツールを」というのは第1回目から放棄してしまい，自分がわからなくて困ったこと，自分がおもしろいと思ったことを，DOSの周辺から引きずり出してきては御託をならべるというスタイルで押し通してしまった．

なにかしらためになる，どこか1つくらいはおもしろいものをと心掛けてきたつもりではあるが，はたしていかがだっただろうか．連載で解説してきたことをヒントになにかいいプログラムを開発される読者がいれば，筆者の喜びこれに優るものはない．

また近いうちに，誌上でお目にかかることもあろうかと思う．そのときはまたよろしくお願いしたい．最後までお付き合いいただいた読者にはもちろんであるが，ヒントや資料を探してくださったり，ときどき電子メールで戯言にお付き合いいただき，また怠け者の筆者を叱咤激励してくださった編集部の鈴木嘉平氏に心からお礼申し上げる．

リスト4 relmem.asm

```
PAGE        60,128

.xlist
include     dosfunc.h
.list

_TEXT       SEGMENT WORD PUBLIC 'CODE'
_TEXT       ENDS

_DATA       SEGMENT WORD PUBLIC 'DATA'
_DATA       ENDS

_STACK      SEGMENT para STACK
_STACK      ENDS

DGROUP      GROUP   _DATA, _STACK
            ASSUME  DS: DGROUP, SS: DGROUP
;
; ========== 定数定義 ========================
;
STACK_SIZE  equu    256
; ============================================
;
_TEXT       SEGMENT
        ASSUME CS:_TEXT

;------------- 「REMMEM.EXE」自身及び取得したメモリブロックの検索
; CY ===> 1 = 発見できなかった
; AX ===> remmemのＰＳＰセグメント
search_remmem  proc
        push    ds
        xor     si,si
        mov     ax,[MCB_top]            ; ＭＣＢの先頭から探し始める
search_loop:
        mov     es,ax
        cmp     byte ptr es:[si],'M'    ; 「M」なら続きがある
        jz      search_ok
        cmp     byte ptr es:[si],'Z'    ; 「Z」なら最終ＭＣＢ
        jnz     error_end               ; それ以外なら異常
search_ok:
        mov     ax,es:[si+1]            ; PSPｾｸﾞﾒﾝﾄを読んで
        or      ax,ax
        jz      next
        push    es
        mov     es,ax
        mov     cx,6                    ; 「REMMEM」の文字列を探す
        push    si
        mov     di,100h
        mov     si,offset remmem_id
   rep  cmpsb
        pop     si
        pop     es
        jnz     next                    ; 見つからなかったら次のブロックへ
        clc                             ; 見つかったらそのMCBｾｸﾞﾒﾝﾄを返す
        pop     ds
        ret
next:
        mov     ax,es
        inc     ax                      ; 次のＭＣＢのアドレス計算
        add     ax,es:[si+3]
        cmp     byte ptr es:[si],'Z'    ; 「Z」を処理したら終了
        jnz     search_loop
error_end:
        stc                             ; 最後まで見つからなかった場合
        pop     ds
        ret
search_remmem  endp

;------------- 発見したREMMEMと取得ﾒﾓﾘﾌﾞﾛｯｸの解放
; CY ===> 1 = 発見できなかった
; AX ===> remmemのＰＳＰセグメント
release_mem proc
        push    ds
        mov     [release_para],ax       ; 解放すべきセグメントを保存
        mov     ax,[MCB_top]
        xor     si,si
release_loop:
        mov     es,ax
        cmp     byte ptr es:[si],'M'    ; 「M」なら続きがある
        jz      release_check
        cmp     byte ptr es:[si],'Z'    ; 「Z」なら最終ＭＣＢ
        jnz     all_over                ; それ以外なら異常
release_check:
        mov     ax,es:[si+1]
        or      ax,ax                   ; 未使用なら無処理
        jz      release_next
        cmp     ax,[release_para]       ; このｾｸﾞﾒﾝﾄが目的のものか？
        jnz     release_next
        push    es
        mov     ax,es
        inc     ax                      ; ならばそのセグメントを解放する
        mov     es,ax
        mov     ah,DOS_MRELEASE
        int     21h
        pop     es
release_next:
        mov     ax,es
        inc     ax                      ; 次のＭＣＢのアドレス計算
        add     ax,es:[si+3]
        cmp     byte ptr es:[si],'Z'    ; 「Z」を処理したら終了
        jnz     release_loop
all_over:
        pop     ds
        ret
release_mem endp

;---------------------------------------------------------------
main    proc    near
        mov     ax,DGROUP
        mov     ds,ax
        mov     ah,DOS_GETLIST
        int     21h
        mov     ax,es:[bx-2]            ; MCBセグメント取得
        mov     [MCB_top],ax
main_loop:
        call    search_remmem           ; REMMEMｾｸﾞﾒﾝﾄを探して
        jc      end_proc
        call    release_mem             ; 見つかったら解放処理
        jmp     short main_loop
end_proc:
        mov     ah,DOS_TERMINATE
        mov     al,0
        int     21h
main    endp
_TEXT   ENDS

_DATA   SEGMENT WORD PUBLIC 'DATA'
remmem_id       db      'REMMEM'
MCB_top         dw      ?
release_para    dw      ?
_DATA   ENDS

_STACK  SEGMENT STACK
                db      STACK_SIZE   dup (?)
_STACK  ENDS
        END     main
```

Illustration:
Masami Nozue

実践C塾

基礎テクニック篇　第6回

ちょっと便利な関数群

二代目塾頭　大野元久

今回で，基礎テクニック篇は最終回となります．そこで，いつもよりもサイズがひかえめだけど，ちょっと便利という関数をいくつか紹介することにします．

farポインタは，8086のすべてのメモリ領域にアクセスすることができるポインタです．スモールモデルにおいても，farポインタを使いたくなることはよくあります．

たとえば，本連載の第8回ではテキストVRAMを扱うためにfarポインタを使いました(リスト1)．ここでは，テキストVRAMをunsigned [80] という配列型へのポインタとすることで，tvram[ly] [lx]のように，垂直・水平位置を指定してVRAMの指定した桁を直接アクセスできるようにしました．

しかし，farポインタにも弱点があります．それは，farポインタをサポートするライブラリがほとんどないという点です．Microsoft C 6.0やBorland C++では，ようやくfarポインタを扱うための文字列操作関数がサポートされましたが，Turbo C++にはありません．また，Borland C++やTurbo C++には，farポインタを使ってファイルの読み書きをする関数などはありません (Microsoft Cには，ファイルハンドルを使ってDOSファンクションを呼び出す関数があります)．

リスト2，3に，farポインタを扱う関数を示します．リスト4にfarポインタを扱うマクロをTurbo C++形式にするためのturboc.hを示します．

これらの関数の仕様は表1のとおりです．

文字列やメモリを操作する関数は，Borland C++やMicrosoft C 6.0には標準ライブラリ関数として用意されています．これらのライブラリ関数は，(おそらく)アセンブラで記述されているため，わざわざCで記述するよりもサイズや速度の面で効率的でしょう．defined (__TURBOC__) && __TURBOC__ <= 0x296という行で，コンパイラがTurbo C++であるかどうか判定し，Turbo C++のときのみコードを有効にするようにしています．

実際には，もっと多くの文字列・メモリ操作の関数があります．これらの機能を実現するのは，読者の方々におまかせします．要するに通常のポインタのかわりにfarポインタを使って，関数を記述すればよいでしょう．

farポインタを使って，ファイルストリームとやりとりをする関数は，どちらの処理系にもありません．ここで用意した関数を使えば，グラフィックスの内容をファイルへ記録したり，ファイルからグラフィックスへ読み出すこともできます(リスト5)．

アプリケーションが使うファイルのバックアップをとったり，既存のファイルと同じ内容のものを一時的に作成したりするため，ファイルをコピーする関数が欲しいときがあります．

リスト6に，ファイルをコピーする関数を示します．通常，ファイルをコピーする場合は，タイムスタンプを元のファイルのものに合わせます．このために，オープン中のファイルハンドルから，タイムスタンプの情報を取得

リスト1

```
static unsigned (far *tvram)[80] = (unsigned (far *)[80])0xA0000000L;
static unsigned (far *avram)[80] = (unsigned (far *)[80])0xA0002000L;
```

リスト2 (farfunc.h)

```
/*
 *  farfunc.h:
 *       farポインタを扱う関数のプロトタイプ宣言
 */
#include <stdio.h>
#include <string.h>

#if defined(__TURBOC__) && __TURBOC__ <= 0x0296

char far * far _fstrcat(char far *dest, const char far *src);
char far * far _fstrchr(const char far *str, int c);
int far _fstrcmp(const char far *s1, const char far *s2);
char far * far _fstrcpy(char far *dest, const char far *src);
char far * far _fstrdup(const char far *str);
size_t far _fstrlen(const char far *str);

void far * far _fmemchr(const void far *str, int c, size_t n);
int far _fmemcmp(const void far *s1, const void far *s2, size_t n);
void far * far _fmemcpy(void far *dest, const void far *src, size_t n);
void far * far _fmemmove(void far *dest, const void far *src, size_t n);
void far * far _fmemset(void far *s, int c, size_t n);

#endif

size_t far _ffread(void far *ptr, size_t size, size_t n, FILE *fp);
size_t far _ffwrite(void far *ptr, size_t size, size_t n, FILE *fp);
```

したり設定するためのgetftime, setftime関数を使います．Microsoft Cには，この関数がないので自分で用意しています．

コピーするためのバッファとして，ヒープ領域から1024バイト(CPYBUFSIZ) を確保しています．通常は，この程度のバッファで十分でしょうが，大きなファイルでは，バッファサイズはもっと大きくしておいたほうがよいかもしれません．

このプログラムを-DTESTオプションをつけてコンパイルすれば，プログラム例として，単一のファイルをコピーする実行ファイルを作成できます．このコマンド(fcopy)は，コピー元のファイル名とコピー先のファイル名を完全に指定しなければなりません．

Turbo C++やBorland C++のgettime，あるいはMicrosoft Cで用意されている_dos_gettimeでは，時刻をあらわす構造体のメンバとして1/100秒をあらわすメンバが用意されていますが，PC-9801ではこの値は常に0となります．これは，IBM-PC用のDOSではDOSファンクションが1/100秒の値(実際の精度はもっと悪い)を返すのに対し，PC-9801用のDOSでは常に0しか返さないためです．

ときとして，こうした細かい精度が必要になることもあります．そこで，

表1　関数の仕様

■文字列操作	
_fstrcat	
機能	文字列の連結
形式	char far * far _fstrcat(char far *dest, const char far *src)；
説明	destの指す文字列の末尾にsrcの指す文字列の内容を追加する．
_fstrchr	
機能	文字の検索
形式	char far * far _fstrchr(const char far *str, int c)；
説明	strの指す文字列から，文字cを検索する．
_fstrcmp	
機能	文字列の比較
形式	int far _fstrcmp(const char far *s1, const char far *s2)；
説明	s1の指す文字列とs2の指す文字列とを比較する．
_fstrcpy	
機能	文字列のコピー
形式	char far * far _fstrcpy(char far *dest, const char far *src)；
説明	srcの指す文字列をdestの指す領域へコピーする．
_fstrdup	
機能	文字列の複製
形式	char far * far _fstrdup(const char far *str)；
説明	新しい領域を確保し，strが指す文字列をコピーする．
_fstrlen	
機能	文字列の長さ
形式	size_t far _fstrlen(const char far *str)；
説明	strの指す文字列の長さを調べる．
■メモリ操作	
_fmemchr	
機能	メモリブロックから文字を検索
形式	void far * far _fmemchr(const void far *str, int c, size_t n)；
説明	strが指すメモリブロックの先頭のnバイトから文字cを探す．
_fmemcmp	
機能	メモリブロックの比較
形式	int far _fmemcmp(const void far *s1, const void far *s2, size_t n)；
説明	s1,s2が指すメモリブロックの先頭のnバイトを比較する．
_fmemcpy	
機能	メモリブロックの転送
形式	void far * far _fmemcpy(void far *dest, const void far *src, size_t n)；
説明	srcが指すメモリブロックからnバイトをdestの指すメモリへ転送する．メモリブロックが重複する場合の動作は未定義．
_fmemmove	
機能	メモリブロックの転送（重複するブロックを認める）
形式	void far* far _fmemmove(void far *dest, const void far *src, size_t n)；
説明	srcが指すメモリブロックからnバイトをdestの指すメモリへ転送する．メモリブロックが重複する場合でも正しく動作する．
_fmemset	
機能	メモリブロックに文字をセット
形式	void far * far _fmemset(void far *s, int c, size_t n)；
説明	sが指すメモリブロックの先頭のnバイトに文字cをセットする．
■ファイルの読み書き	
機能	ストリームからバッファへの読み込み
形式	size_t far _ffread(void far *ptr, size_t size, size_t n, FILE *fp)；
説明	ファイルストリームからptrが指すバッファ領域へ，データを読み込む．
機能	バッファからストリームへの書き込み
形式	size_t far _ffwrite(void far *ptr, size_t size, size_t n, FILE *fp)；
説明	ptrが指すバッファ領域から，ファイルストリームへデータを書き込む．

ワンラインNEWS
カシオ計算機はワープロの品揃えを強化した．10万円台の低価格機から，同社としては初の20万円以上の高級機を投入し，シェアの奪還を狙う．

リスト3 (farfunc.c)

```c
/*
 *  farfunc.c:
 *      farポインタを扱う関数
 */
#include <string.h>
#include <stdlib.h>
#include "turboc.h"
#include "farfunc.h"

#if defined(__TURBOC__) && __TURBOC__ <= 0x0296

/*
 *  文字列の連結
 */
char far * far _fstrcat(char far *dest, const char far *src)
{
    char far *p, far *q;

    p = dest;
    q = src;
    while (*p)
        ++p;
    do {
        *p++ = *q;
    } while (*q++);
    return dest;
}

/*
 *  文字の検索
 */
char far * far _fstrchr(const char far *str, int c)
{
    while (*str) {
        if (*str == (char)c) return str;
        ++str;
    }
    return NULL;
}

/*
 *  文字列の比較
 */
int far _fstrcmp(const char far *s1, const char far *s2)
{
    while (*s1 || *s2) {
        if (*s1 != *s2) return (unsigned char)*s1 - (unsigned char)*s2;
        ++s1;
        ++s2;
    }
    return 0;
}

/*
 *  文字列のコピー
 */
char far * far _fstrcpy(char far *dest, const char far *src)
{
    char far *p;

    p = dest;
    do {
        *p++ = *src;
    } while (*src++);
}

/*
 *  文字列の長さ
 */
size_t far _fstrlen(const char far *str)
{
    size_t n;

    for (n = 0; *str; str++)
        n++;
    return n;
}

/*
 *  文字列の複製
 */
char far * far _fstrdup(const char far *str)
{
    char *p;

    p = malloc(_fstrlen(str));
    if (p) _fstrcpy(p, str);
    return (char far *)p;
}

/*
 *  メモリブロックから文字を検索
 */
void far * far _fmemchr(const void far *str, int c, size_t n)
{
    unsigned char far *p;

    for (p = (unsigned char far *)str; n; p++, n--)
        if (*p == (char)c) return p;
    return NULL;
}

/*
 *  メモリブロックの比較
 */
int far _fmemcmp(const void far *s1, const void far *s2, size_t n)
{
    unsigned char far *p, far *q;

    for (p = (unsigned char far *)s1, q = (unsigned char far *)s2;
            n; n--, p++, q++)
        if (*p != *q) return *p - *q;
    return 0;
}

/*
 *  メモリブロックの転送
 */
void far * far _fmemcpy(void far *dest, const void far *src, size_t n)
{
    unsigned char far *p, far *q;

    for (p = (unsigned char far *)s1, q = (unsigned char far *)s2;
            n; n--, p++, q++)
        *p = *q;
    return dest;
}

/*
 *  メモリブロックの転送（重複するブロックを認める）
 */
void far * far _fmemmove(void far *dest, const void far *src, size_t n)
{
    unsigned char far *p, far *q;

    p = (unsigned char far *)s1;
    q = (unsigned char far *)s2;
    if ((unsigned char huge *)p < (unsigned char huge *)q) {
        while (n--)
            *p++ = *q++;
    } else {
        p += n;
        q += n;
        while (n--)
            *--p = *--q;
    }
    return dest;
}

/*
 *  メモリブロックに文字をセット
 */
void far * far _fmemset(void far *s, int c, size_t n)
{
    unsigned char far *p;

    p = s;
    while (n--)
        *p++ = (unsigned char)c;
    return s;
}

#endif

/*
 *  ストリームからバッファへの読み込み
 */
size_t far _ffread(void far *ptr, size_t size, size_t n, FILE *fp)
{
    char buf[BUFSIZ];
    unsigned char huge *p;
    size_t len;
    long num = (long)size * n;
    long total = 0;

    p = (unsigned char huge *)ptr;
    while (num > 0L) {
        len = (num >= BUFSIZ) ? BUFSIZ : (int)num;
        len = fread(buf, 1, len, fp);
        movedata(FP_SEG(buf), FP_OFF(buf), FP_SEG(p), FP_OFF(p), len);
        total += len;
        p += len;
        num -= len;
        if (len < BUFSIZ) break;
    }
    return (size_t)(total / size);
}
/*
 *  バッファからストリームへの書き込み
 */
size_t far _ffwrite(void far *ptr, size_t size, size_t n, FILE *fp)
{
    char buf[BUFSIZ];
    unsigned char huge *p;
    size_t len;
    long num = (long)size * n;
    long total = 0;

    p = (unsigned char huge *)ptr;
    while (num > 0L) {
        len = (NUM >= BUFSIZ) ? BUFSIZ : (int)num;
        len = fread(buf, 1, len, fp);
        movedata(FP_SEG(buf), FP_OFF(buf), FP_SEG(p), FP_OFF(p), len);
        total += len;
        p += len;
        num -= len;
        if (len < BUFSIZ) break;
    }
    return (size_t)(total / size);
}
```

PC-9801のタイマBIOSを利用して1/100秒単位での時間を測定するための関数を用意しました．リスト7にプログラムを示します．このプログラムもturboc.hをインクルードしています．

Microsoft Cでは，タイマBIOSを扱う関数が用意されていないので，Turbo C++／Borland C++のpc98timerに相当する関数を用意しました．

このプログラムをコンパイルするときは，スタックチェックを外してコンパイルしなければならないことに注意してください．コマンドラインコンパイラを使っている場合，Turbo C++／Borland C++では-N-で，Microsoft Cでは/Gsで，スタックチェックを解除できます．とくに，Microsoft C 6.0では，デフォルトでスタックチェックが有効になっていますので，必ずこのオプションを指定するようにしてください．

割り込み関数が呼び出されるときに使われるスタックは，ハードウェア割り込みが発生した時点で使われていたスタックそのものです．したがって，たとえばDOSファンクションの処理中に割り込みが発生すれば，DOSファンクションが使っていたスタックがスタックチェックの対象となります．もちろん，これでは正常なスタックチェックが行われませんので，意味のないStack overflow!などのメッセージが表示されてしまうことにもなります．

プログラムの確保したスタック以外のスタックが使われているために，発生している別の問題もあります．Microsoft Cのために用意したpc98timer関数では，int86xでなく_asm文を使用しています．pc98timerは，その性質上割り込みハンドラから呼び出されることがありますが，int86xはDS==SSであることを前提にプログラミングされています．pc98timerがint86xを使えなかったのは，こうした割り込み関数から呼び出すことができなかったためです．

Turbo C++／Borland C++も，Microsoft C 6.0も，interrupt型を指定した関数では，自動的にスタックチェックを外してくれるようですが，そこから呼び出される「普通の」関数のスタックチェックは外されません．ちなみに，いずれの処理系も#

リスト4 (turboc.h)

```
/*
 *  Microsoft C のマクロ、構造体、関数の定義を Turbo C++ 形式に変更する。
 */
#ifndef __TURBOC_H
#define __TURBOC_H

#if !defined(__TURBOC__)
/*
 *  ｆａｒポインタ用のマクロ
 */
#undef FP_SEG
#undef FP_OFF
#define FP_SEG(fp) ((unsigned)((unsigned long)(fp) >> 16))
#define FP_OFF(fp) ((unsigned)(fp))
#define MK_FP(seg,ofs) (((unsigned long)(unsigned)(seg)<<16) | (unsigned)(ofs))
#endif

#endif  /* __TURBOC_H */
```

リスト5 (gfile.c)

```
/*
 *  gfile.c:
 *      グラフィックスＶＲＡＭ←→ファイル転送
 */
#include <stdio.h>
#include "turboc.h"
#include "farfunc.h"

/*
 *  グラフィックスイメージ全体をファイルへ記録
 */
int gsave(const char *file)
{
    FILE *fp;
    size_t n;

    fp = fopen(file, "wb");
    if (fp == NULL) return EOF;

    n = _ffwrite(MK_FP(0xA800, 0), 1, 32000, fp);
    if (n != 32000) {
        fclose(fp);
        return EOF;
    }
    n = _ffwrite(MK_FP(0xB000, 0), 1, 32000, fp);
    if (n != 32000) {
        fclose(fp);
        return EOF;
    }
    n = _ffwrite(MK_FP(0xB800, 0), 1, 32000, fp);
    if (n != 32000) {
        fclose(fp);
        return EOF;
    }
#if 0
/* 輝度プレーンも保存する場合 */
    n = _ffwrite(MK_FP(0xE000, 0), 1, 32000, fp);
    if (n != 32000) {
        fclose(fp);
        return EOF;
    }
#endif
    fclose(fp);
    return 0;
}

/*
 *  グラフィックスイメージ全体をファイルから読み出す
 */
int gload(const char *file)
{
    FILE *fp;
    size_t n;

    fp = fopen(file, "rb");
    if (fp == NULL) return EOF;

    n = _ffread(MK_FP(0xA800, 0), 1, 32000, fp);
    if (n != 32000) {
        fclose(fp);
        return EOF;
    }
    n = _ffread(MK_FP(0xB000, 0), 1, 32000, fp);
    if (n != 32000) {
        fclose(fp);
        return EOF;
    }
    n = _ffread(MK_FP(0xB800, 0), 1, 32000, fp);
    if (n != 32000) {
        fclose(fp);
        return EOF;
    }
    n = _ffread(MK_FP(0xE000, 0), 1, 32000, fp);
    if (n != 0 || n != 32000) {
        fclose(fp);
        return EOF;
    }
    fclose(fp);
    return 0;
}
```


ワンラインNEWS
松下電器産業はDTPシステムの中級機「パナページZ600」を発売した．DTPシステムでは初めて住所録ソフトウェアを内蔵し，ワープロ感覚で使えるのが特徴．

pragmaでスタックチェックを制御することができます.

リスト6 (fcopy.c)

```
/*
 *  fcopy.c:
 *      ファイルをコピーするモジュール
 */
#include <stdarg.h>
#include <stdlib.h>
#include <io.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

#if !defined(__TURBOC__)
#include <dos.h>

struct  ftime   {
    unsigned    ft_tsec  : 5;   /* Two second interval */
    unsigned    ft_min   : 6;   /* Minutes */
    unsigned    ft_hour  : 5;   /* Hours */
    unsigned    ft_day   : 5;   /* Days */
    unsigned    ft_month : 4;   /* Months */
    unsigned    ft_year  : 7;   /* Year */
};

int getftime(int handle, struct ftime *ftimep)
{
    union REGS regs;

    regs.x.ax = 0x5700;
    regs.x.bx = handle;
    intdos(&regs, &regs);
    if (regs.x.cflag) return -1;
    ((unsigned *)ftimep)[0] = regs.x.cx;
    ((unsigned *)ftimep)[1] = regs.x.dx;
    return 0;
}

int setftime(int handle, struct ftime *ftimep)
{
    union REGS regs;

    regs.x.ax = 0x5701;
    regs.x.bx = handle;
    regs.x.cx = ((unsigned *)ftimep)[0];
    regs.x.dx = ((unsigned *)ftimep)[1];
    intdos(&regs, &regs);
    return regs.x.cflag ? -1 : 0;
}

#endif

#define CPYBUFSIZ 1024  /* コピー用のバッファサイズ */

/*
 *  ファイルコピー
 */
int filecopy(const char *source, const char *dest)
{
    char *cbuf;
    int fds, fdd;
    size_t size;
    struct ftime stamp;

    fds = open(source, O_RDONLY | O_BINARY);
    if (fds < 0)
        return -1;
    fdd = open(dest, O_CREAT | O_TRUNC | O_WRONLY | O_BINARY,
                        S_IREAD | S_IWRITE);
    if (fdd < 0) {
        close(fds);
        return -1;
    }
    cbuf = malloc(CPYBUFSIZ);
    if (cbuf == NULL) {
        close(fds);
        close(fdd);
        unlink(dest);
        return -1;
    }
    while ((size = read(fds, cbuf, CPYBUFSIZ)) > 0) {
        if (write(fdd, cbuf, size) < size)
            break;
    }
    getftime(fds, &stamp);
    setftime(fdd, &stamp);
    free(cbuf);
    close(fds);
    close(fdd);
    return size == 0 ? 0 : -1;
}

#if defined(TEST)
#include <stdio.h>

int main(int argc, char **argv)
{
    if (argc < 3) {
        fprintf(stderr, "fcopy source dest¥n");
        return 1;
    }
    return filecopy(argv[1], argv[2]) ? 1 : 0;
}
#endif
```

リスト7 (ticks.c)

```
#include <stddef.h>

#if defined(__TURBOC__)
#include <pc98.h>
#else
#include <dos.h>
#include "turboc.h"

static void (*timerfunc)(void) = NULL;
static volatile int timerdone = 1;

void interrupt far timer_handler()
{
    void (*tfunc)(void) = timerfunc;

    timerfunc = NULL;
    timerdone = 1;
    if (tfunc) (*tfunc)();
}

void (*pc98timer(unsigned interval, void (*fptr)()))()
{
    void (*prevfunc)() = timerfunc;
    unsigned fseg, foff;

    timerdone = 0;
    timerfunc = fptr;
    if (fptr == NULL) interval = 1;
    _disable();
    fseg = FP_SEG(timer_handler);
    foff = FP_OFF(timer_handler);
    _asm {
        mov cx, interval
        mov bx, foff
        mov ax, fseg
        mov es, ax
        mov ah, 2
        int 1Ch
    }
    _enable();
    if (fptr == NULL) {
        while (!timerdone)
            ;
    }
    return prevfunc;
}
#endif

volatile long ticks = 0;

void getticks(void)
{
    ++ticks;
    pc98timer(1, getticks);
}

void timerstart(void)
{
    ticks = 0;
    pc98timer(1, getticks);
}

void timerstop(void)
{
    pc98timer(0, NULL);
}

#if defined(TEST)
#include <stdio.h>
#include <conio.h>

void main()
{
    timerstart();
    while (!kbhit())
        printf("ticks = %ld¥r", ticks);
    timerstop();
    printf("¥n");
    getch();
}
#endif
```

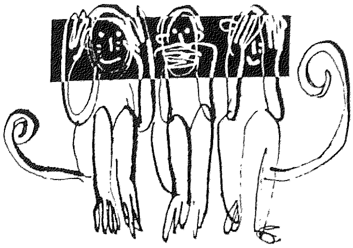
システムの状態を参照する関数

Turbo C++／Borland C++には，pc98equip，Microsoft C 6.0には_bios_equiplistという関数が用意されていて，それぞれシステムの設定や周辺機器に関する情報を取得できるようになっています．

これらは，たいていBIOSのワークエリアなどを参照しているだけですが，返される値の性質がいくぶん違いますし，いちいちすべての内容を取得することはありません．そこで，いくつかの性質についてメモリ内のワークエリアから，環境を取得できるマクロを定義してみました(リスト8)．これをヘッダファイルとしてインクルードすれば，表2のマクロを使うことができます．

さいごに

今回で私の担当は最後となります．最後までおつきあいくださった読者の皆様，どうもありがとうございました．基礎テクニック篇といいながら，プログラムリストの紹介が多くて飽きてしまわれた方，どうもすみませんでした．この連載で紹介したプログラムが少しでも皆様のプログラム開発の手助けとなれば幸いです．では．



リスト8 (equip.h)

```
#include <dos.h>
#include "turboc.h"

/* 特殊キーの押下状態を返す */
#define shiftkey() (*(unsigned char far *)MK_FP(0x0000, 0x053A))

/* shiftkey が返す値に対応するビットをあらわす識別子 */
#define KB_SHIFT 0x01
#define KB_CAPS 0x02
#define KB_KANA 0x04
#define KB_GRPH 0x08
#define KB_CTRL 0x10

/* ハイレゾリューションモードの場合は、0でない値を返す */
#define ishires() (*(unsigned char far *)MK_FP(0x0000, 0x0501) & 0x08)

/* グラフィックスが表示されていれば0でない値を返す */
#define isgraphon() (*(unsigned char far *)MK_FP(0x0000, 0x054C) & 0x80)

/* カーソルの水平位置（基点は0） */
#define cursorx() ((int)*(unsigned char far *)MK_FP(0x0060, 0x011C))

/* カーソルの垂直位置（基点は0） */
#define cursory() ((int)*(unsigned char far *)MK_FP(0x0060, 0x0110))

/* 画面の行数を返す */
#define crtlines() ((int)*(unsigned char far *)MK_FP(0x0060, 0x0112) + 1)

/* ファンクションキーの表示状態 */
#define isfkeyon() ((int)*(unsigned char far *)MK_FP(0x0060, 0x0111))
```

表2 マクロ

機能	特殊キーの押下状態
形式	unsigned char shiftkey (void) ;
説明	特殊キーの押下状態を調べる．返された値とKB_SHIFT ([SHIFT])，KB_CAPS ([CAPS])，KB_KANA ([カナ])，KB_GRPH ([GRPH])，KB_CTRL ([CTRL]) との論理積をとると，該当するキーが押されていれば0でない値になる．
機能	ハイレゾリューションモードの判別
形式	int ishires (void) ;
説明	ハイレゾリューションモードであれば0でない値が，そうでなければ0が返される．
機能	グラフィックスの表示状態
形式	int isgraphon (void) ;
説明	グラフィックスが表示されていれば0でない値が返される．
機能	カーソルの水平位置
形式	int cursorx (void) ;
説明	DOSの標準出力が指しているカーソルの水平位置を返す．基点は0．
機能	カーソルの垂直位置
形式	int cursory (void) ;
説明	DOSの標準出力が指しているカーソルの垂直位置を返す．基点は0．
機能	テキスト画面の行数
形式	int crtlines (void) ;
説明	テキスト画面の行数を返す．ファンクションキーが表示されているときは，1行少ない値が返される．
機能	ファンクションキーの表示状態
形式	int isfkeyon (void) ;
説明	ファンクションキーが表示されていれば0でない値が返される．

ワンラインNEWS
三洋電機は洗濯機や掃除機など家電製品の制御手法としてニューロとファジー制御のどちらを採用すると有利かを簡単に比較できるエキスパートシステムを開発した．

98活用アセンブラ技法

タイマ編
第2回

時間取得・表示プログラム/プリンタスプーラの作成

星野 操

前回は，PC-9801のタイマBIOSを拡張して，マルチイベントが処理可能なようにした．ファンクションはハイレゾモードのBIOS呼び出しを参考にしたが，完全な互換性は取らずに拡張した．
今回は，このタイマBIOSを利用したプログラムをBIOS利用のサンプルとして2つほど作成した．

res2.asmの改良

常駐プログラムの雛型作成

前回timerを常駐プログラムとして構成するために，以前に掲載したresident. asmを手直しして，res2. asmを作成しました．今回，さらにこれに多少手を入れて，一般的な常駐プログラムに便利なようにします．

まず，今回作成したres3. asmにおける変更部分をリスト1に示します．変更点は，主要なポイントで外部ルーチンを呼び出せるようにした，というものです．具体的には，INT2Fのインターフェースルーチンの中でリリース機能が呼び出された場合に，release_funcを呼び出すようにします．割り込みベクタに関しては，res3側のほうで全部処理されます．しかし，割り込み以外の方法でチェインを形成したりしている場合には，正常にリリース処理ができません．

たとえば，今回はマルチイベントのタイマを利用します．このタイマ用のワークエリアは自前のリンク構造を取っていますので，自動的には解放されません．リリース時に明示的に解放処理をしてやる必要があります．release_funcという手続きの中でこのような処理をするように記述しておけばよいのです．

check_func

次のcheck_funcは，常駐を開始する前に呼び出される関数です．たとえば，今回使用するプログラムではtimerの常駐を必要とします．このようなときに，check_funcの中で常駐チェックを行えば，timerがあらかじめ常駐していないときにエラーで中断することができるのです．ここで，AXが0であればそのまま常駐処理が継続され，非0であれば，返り値1をもって中断します．中断時にはとくにメッセージが出力されないので，必要な場合にはあらかじめこれをプリントするようにします．なお，とくにこのようなチェックが必要ない場合には，AXに0を設定してそのままリターンします．

intf_func

最後に追加されたintf_funcは，インターフェース割り込みINT2Fの機能を拡張するためのものです．デフォルトでは，常駐チェックの0と領域解放の1だけがサポートされていますが，プログラムによってはそのほかのインターフェース機能が要求されることがあります．

機能コードを示すBXレジスタが0/1でない場合には，このintf_funcが呼び出されます．必要なら，このルーチンの中でBXの値をチェックして処理をすることが可能となります．なお，BXの値はあらかじめ-2されているので注意する必要があります．

以上の機能を追加するために，リスト1に示される変更をほどこして，res3. asmを作成し，レジデントプログラムのスケルトンとして使用します．

リスト1　res2.asmからの変更点

```
*** res2.asm    Thu Sep 05 20:31:30 1991
--- res3.asm    Wed Oct 09 22:50:46 1991
***************
*** 22,28
            extrn   my_id:word
            extrn   interrupt_list:byte
            extrn   init_func:far

  int_work          struc
  int_num           dw      ?

--- 22,31 -----
            extrn   my_id:word
            extrn   interrupt_list:byte
            extrn   init_func:far
+           extrn   release_func:near
+           extrn   check_func:far
+           extrn   intf_func:near

  int_work          struc
  int_num           dw      ?
***************
*** 64,70
            dec     bx
            @e_if   z
                    ; release program
                    push    ds

                    ; resume all vector

--- 67,74 -----
            dec     bx
            @e_if   z
                    ; release program
+                   call    release_func
                    push    ds

                    ; resume all vector
***************
*** 101,107

                        ; return psp to release me
                        mov     dx,$psp         ; $psp is a system variable
                @endif

                pop     ds                      ; resume ds value

--- 105,113 -----

                        ; return psp to release me
                        mov     dx,$psp         ; $psp is a system variable
+               @else
+                       call    intf_func       ; extended function
                @endif

                pop     ds                      ; resume ds value
***************
*** 259,265
                        terminate <1>           ; error return
                @endif

                ; install vector
                push    ds
                mov     cx,_TEXT

--- 265,278 -----
                        terminate <1>           ; error return
                @endif

+               ; check condition
+               call    check_func
+               or      ax,ax
+               @if     nz
+                       terminate <1>           ; error
+               @endif
+
                ; install vector
                push    ds
                mov     cx,_TEXT
```


ワンラインNEWS
米クレイ・リサーチは計算速度が同社の最上級製品の約24倍で，本体部分の大きさが約1/50のスーパーコンピュータ「トライトン（仮称）」の開発に着手した．

dispclk.asm
時間の取得と表示

次に示すdispclk. asmはtimerと共用して，画面下部に時間を表示するものです．通常は時・分を表示しますが，シフトキーを押すことによって秒・1/100秒を表示するようにします．

時間を取得するには，DOSコールやBIOSコールが使用できます．DOSコールを使用しても，結局はBIOSコールが実行されます．ここで，任意の時点でBIOSコールを実行すると，すでにBIOSの中にはいっている場合には，リアルタイムクロックの設定がおかしくなることがあります．ですから，任意の時点でこのシステムの時間を取得することはできません．

前回説明したように，timerでは長期的にかなり正確な間隔の割り込みを発生させます．ですから，これを計時の契機として使用することができます．いちばん最初の常駐のときにDOSコールで時間を取得し，それから10msの割り込みごとにカウンタをクロックアップします．そして，それに基づいて時間を画面に直接書き込めばよいわけです．

プロローグ部分

プログラムは，まず最初にtimerと同様なプロローグ部分があります．public手続きは，前記のように外で処理するルーチンが追加されたために，3つ増えています．my_idを利用するほかに，timerの常駐チェックのためのtimerのIDも知っておく必要があります．int_workの構造体などもtimerと同様ですが，独自に使用する割り込みはないので，interrupt_listの中身は空です．

timer_work構造体

timer_work構造体はtimerで定義されているものと同一で，この構造体を1つ用意しておきます．そのほかに，時分秒カウンタと，前記timer非常駐時に出力するメッセージを用意しておきます．

ascii_value

最初に定義されている関数ascii_valueは，0〜99までの数値を，2桁のアスキー文字に変換するものです．AAAなどの命令を使えばもっと簡単になるでしょうが，ここでは単純に10

リスト2　res3.asm

```
;------------------------------
; res2: resident handling part
;
; $Log: RCS/res2.asm $
; revision 1.1 TAN 91/09/05 07:31:30
; Initial revision
;
;
;------------------------------

        dosseg
        .model  small,pascal

        masm51                          ; delete if masm

        include condx.inc

        include utils.inc
        include dosfile.inc

        .code
        extrn   my_id:word
        extrn   interrupt_list:byte
        extrn   init_func:far
        extrn   release_func:near
        extrn   check_func:far
        extrn   intf_func:near

int_work        struc
int_num         dw      ?
int_func        dw      ?
int_old         dd      ?
int_work        ends

;------------------------------
; int2f: mulitiplexed interrupt entry
; argument:
;       AX: ID
;       BX: function code
;          0 - check residence
;          1 - release program
; return:
;       DX: psp of program (function 0/1)
;       AX: broken

        .code
vect2f  dd      ?

_int2f:
        ; first, check interface ID
        cmp     ax,cs:my_id
        @if     nz
                jmp     vect2f          ; chain to old interrupt
        @endif

        sti                             ; enable interrupt
        push    ds
        mov     ax,_DATA                ; set data segment value
        mov     ds,ax

        ; branch with function code
        or      bx,bx
        @if     z
                ; check residence, return psp
                mov     dx,$psp         ; $psp is a system variable
        @else
        dec     bx
        @e_if   z
                ; release program
                call    release_func
                push    ds

                ; resume all vector
                mov     bx,offset interrupt_list
                @begin
                        mov     ax,cs:[bx].int_num
                        lds     dx,cs:[bx].int_old
                        add     bx,size int_work

                        or      ax,ax
                @while  nz
                        dos     <25h,,,>
                @repeat

                ; release int2f vector
                lds     dx,vect2f
                mov     al,2fh
                dos     <25h,,,>

                pop     ds

                ; then free environment
                mov     es,$psp         ; first, get environ seg
                mov     ax,es:[2ch]
                mov     bx,ax
                dec     ax
                mov     es,ax
                mov     ax,es:[1]       ; check owner of environ seg
                cmp     ax,$psp
                @if     z
                        ; environment is not released
                        mov     es,bx
                        dos     <49h,,,>        ; free environ
                @endif

                ; return psp to release me
                mov     dx,$psp         ; $psp is a system variable
        @else
                call    intf_func       ; extended function
        @endif
```

```
        pop     ds                      ; resume ds value
        iret

;----------------------------
; initialize routines
; define intialize code segment
;

        .data
end_pos db      0                       ; show the end of data segment
        public  end_pos

r_option        db      '-r'            ; option string definition
dup_message     db      'program is already installed',0dh,0ah,0
not_res_message db      'program not installed',0dh,0ah,0
rel_ok_message  db      'program correctly released',0dh,0ah,0
correct_message db      'program correctly installed',0dh,0ah,0
arg_err_message db      'argument range error',0dh,0ah,0
vect23  dd      ?                       ; for ctrl_c vector

init_seg        segment 'stack'         ; special segment for initialize
        assume  cs:init_seg             ; cs: init_seg

;----------------------------
; strncmp: compare two strings
; arguent:
;       p1, p2: buffer address
;       n: number to compare
; return:
;       ax: result, 0 for equall, nz for not equall

func    strncmp <p1, p2, n> <si di>
        mov     dx,0                    ; dx for flag
        mov     si,p1
        mov     di,p2
        push    ds                      ; setup es
        pop     es

        @begin
                cmp     n,0
        @while  nz
                lodsb                   ; compare characters
                scasb
                @if     nz
                        mov     dx,1    ; non zero value
                        @break
                @endif

                or      al,al           ; check string tail
                @if     z
                        @break
                @endif
                dec     n               ; decriment counter
        @repeat

        mov     ax,dx                   ; set return value
        ret
endf

;----------------------------
; tominate: terminate function
; argument:
;       status: exit status
;
func    terminate       <status>
        ; restore int23 vector
        lds     dx,vect23
        mov     al,23h
        dos     <25h,,,>

        ; terminate
        mov     ax,status
        dos     <4ch,,,>
endf

dummy_int:
        iret

;----------------------------
; initial: initialize part
;

initial:
        @setup                          ; parse command line

        ; disable ^c interrupt
        mov     al,23h                  ; get ^c vector
        dos     <35h,,,>
        mov     word ptr vect23,bx      ; save it
        mov     word ptr vect23+2,es

        push    ds                      ; set dummy interrupt entry
        push    cs
        pop     ds
        mov     dx,offset dummy_int
        dos     <25h,,,>
        pop     ds

        push    ds
        mov     cx,_TEXT
        mov     ds,cx
        assume  ds:_TEXT
        mov     ax,my_id                ; then check residence
        pop     ds
        assume  ds:_DATA

        mov     bx,0
        mov     dx,bx
        int     2fh

        ; duplicate check
        or      dx,dx
        @if     nz                      ; already installed
                ; check R option
                mov     bx,_argv
                strncmp <offset r_option,[bx+2],2>      ; compare option
                @if     nz              ; reinstall error
                        mov     dx,offset dup_message
                        @puts <dx>

                        terminate <1>   ; error code
                @endif

                ; release program
                push    ds
                mov     cx,_TEXT
                mov     ds,cx
                assume  ds:_TEXT
                mov     ax,my_id
                pop     ds
                assume  ds:_DATA
                mov     bx,1
                int     2fh

                ; and release program area
                mov     es,dx
                dos     <49h,,,>

                ; message
                mov     dx,offset rel_ok_message
                @puts   <dx>

                ; terminate successfully
                terminate <0>
        @endif

        ; check -r flag
        mov     bx,_argv
        strncmp <offset r_option,[bx+2],2>
        @if     z
                mov     dx,offset not_res_message       ; not yet installed
                @puts   <dx>

                terminate <1>           ; error return
        @endif

        ; check condition
        call    check_func
        or      ax,ax
        @if     nz
                terminate <1>           ; error
        @endif

        ; install vector
        push    ds
        mov     cx,_TEXT
        mov     ds,cx
        assume  ds:_TEXT

        mov     al,2fh                  ; save 2fh vector
        dos     <35h,,,>
        mov     word ptr vect2f,bx
        mov     word ptr vect2f+2,es

        mov     dx,_TEXT                ; set 2fh vector
        mov     ds,dx
        mov     dx,offset _int2f
        dos     <25h,,,>

        ; and install other interrupts
        mov     si,offset interrupt_list
        @begin
                mov     ax,[si].int_num
                or      ax,ax
        @while  nz
                dos     <35h,,,>
                mov     word ptr [si].int_old,bx
                mov     word ptr [si].int_old+2,es

                mov     dx,[si].int_func
                dos     <25h,,,>

                add     si,size int_work
        @repeat

        pop     ds
        assume  ds:_DATA


        ; message output
        mov     dx,offset correct_message
        @puts <dx>

        ; call sub function
        call    init_func

        ; stay resident
        mov     dx,offset dgroup:end_pos        ; calculate paragraph size
        add     dx,15                   ; data segment length
        mov     cl,4
        shr     dx,cl

        add     dx,_DATA                ; add data segment offset
        sub     dx,$psp

        mov     al,0                    ; success flag
        dos     <31h,,,>                ; keep process

init_seg        ends

        end     initial
```


ワンラインNEWS
米シーゲート・テクノロジーは1インチ厚の3.5インチとしては最大の記憶容量をもつHDD「ST3600シリーズ」を開発，年末にかけて発売する．

で割ってから'0'コードをたすことによって文字に変換しています．ALに上位桁の数値，AHに下位桁の数値が入ります．これは，単純にfuncマクロで定義されています．

interrupt_func

その次のinterrupt_funcは，INT1CHのタイマ割り込みサービスによって起動される割り込み関数です．これは，INTERRUPT形式のルーチンなので，ラベルを利用して定義しています．

まずDSの値を設定したあと，タイマカウンタをアップデートします．それから，下位メモリのシフトステータスをチェックすることによって，シフトキーが押されているかどうかを判断します．そして，シフトキーの状態に依存して時・分か秒・msの値を取得します．

それをascii_valueを利用してコード変換したあと，画面の左下隅に書き込みます．それからレジスタを復元し，AHに0FFHを設定してからIRETで復帰します．AHが0だとリピートが打ち切られるので，注意しなければいけません．

release_func

次のrelease_funcは，常駐解放時にタイマサービスを中止するものです．ここでは，AH＝3のサービスを利用し，ワークエリアをリンクトリストから外します．それから画面端の時間表示をクリアし，復帰します．そのほかの常駐処理は，すべてres3の中で処理されます．

このルーチンは，とくにインターフェース機能を利用しないので，intf_funcは空になっています．時間表示を24H／12H切り替えできるようにする場合には，このintf_funcを利用することで可能となります．なお，ここではfuncマクロではなく通常のprocを使用しています．これは，ラベル名の自動変換を避けるためのものです．check_funcなどがやはりprocで実現されているのも，同一の理由によるものです．

check_func／init_func

次のcheck_funcとinit_funcは起動時のみ使用されるものです．ですから起動時のセグメントinit_segに設定します．とくにpublic宣言はしていない

リスト3　dispclk.asm

```
;---------------------------
; dispclk: display time
;
; $Log: RCS/timer.asm $
;
;---------------------------

        dosseg
        .model  small,pascal

        include condx.inc
        include utils.inc

        masm51

        .stack  2000h

; for interrupt
        .code
        public  my_id,interrupt_list,init_func,release_func
        public  check_func,intf_func

my_id   dw      2585h
timer_id        equ     2584h

int_work        struc
int_num         dw      ?
int_func        dw      ?
int_old         dd      ?
int_work        ends

interrupt_list  label   byte
        dw      0

        .data
timer_work      struc
forward_link    dd      ?               ; forward link
timer_value     dw      ?               ; decrement count

timer_count     dw      ?               ; timer work initial value
timer_routine   dd      ?               ; execute routine on timeout

backward_link   dd      ?               ; backword link pointer
timer_status    db      ?               ; overflow flag, repeat mode
timer_divide    db      ?               ; timer divide work
timer_work      ends

my_work timer_work      <>

hour    db      ?
minute  db      ?
second  db      ?
tmsec   db      ?

no_timer_mes    db      'timer routine does not resident!',0dh,0ah,0

        .code

func    ascii_value     <data>
        mov     ax,data
        mov     dl,10
        div     dl
        add     ax,3030h
        ret
endf

interrupt_func:
        ; register save, DS set
        push    ds
        push    dx
        mov     ax,_DATA
        mov     ds,ax

        ; first increment time
        inc     tmsec
        cmp     tmsec,100
        @if     ae
                mov     tmsec,0
                inc     second
                cmp     second,60
                @if     ae
                        mov     second,0
                        inc     minute
                        cmp     minute,60
                        @if     ae
                                mov     minute,0
                                inc     hour
                                cmp     hour,24
                                @if     ae
                                        mov     hour,0
                                @endif
                        @endif
                @endif
        @endif

        ; and display it
        ; check shift key value

        mov     ax,0
        mov     ds,ax
        test    byte ptr ds:[53ah],1
        mov     ax,_DATA
        mov     ds,ax
        @if     z
                ; not shifted
                ; get hour/minute
                mov     al,minute
                mov     ah,hour
        @else
                ; shifted
                ; get second/tmsec
                mov     al,tmsec
                mov     ah,second
        @endif

        ; convert second data into ascii
        push    ax
        mov     ah,0
```

ワンラインNEWS
ジャパンマクニクスはノート型パソコン用のLAN接続機器「Note EXOS」を開発した．同機はパソコン本体の下に置け，無駄な場所を必要としない．

ので，res3のセグメントとまとめられることはありませんが，これらの手続きはfarタイプと宣言されているので，呼び出しに問題が生じることはありません．ただし，linkを使用するとうまくこのセグメントは最後に回らないようです．

check_funcは，まずtimer_idを用いてtimerの常駐チェックを実行します．もし常駐していなければ，メッセージを出力してAXに非０の値を設定します．そうでなければAXをクリアします．これで常駐処理を継続するかどうかを指定するわけです．

最後のinit_funcは，まずタイマ用のワークエリアに値を設定して，それからDOSの時間取得の機能を呼び出し，それでタイマカウンタを初期化します．最後にint1Chのタイマ割り込みを起動して終了します．

アセンブルの方法

このプログラムを作成するには，dispclkとres3を別々にアセンブルし，最後にdispclkとres3をこの順番でリンクすれば実行可能ファイルが作成されます．timerを常駐させてからdispclkを常駐させることによって，画面左下に時間が表示されるようになります．

プリンタスプーラの作成

2つの機能が必要

さて，その次に簡単なプリンタスプーラを作ってみることにします．プリンタスプーラは，DOS標準でprint.com／exeが付属してきますし，そのほかにも各種RAMボードなどに添付されるドライバにも，バンクメモリやプロテクトメモリ，EMSをバッファ領域に使用するスプーラがついてきます．

これらは多少相違があり，前者はすでに作成されているファイルをプリンタに打ち出すものですが，後者はプリンタのBIOSレベルで出力をトラップし，とくにファイル出力に対応していないプログラムでもスプーリングが効くようになっています．

後者の場合でもCOPY FILE PRNと実行することによってスプーリングされますが，スプーラに割り当てられたメモリよりプリントするファイルが大きかった場合には，実行が中断します．前者は，ファイルを作らないといけないという問題はあります

```
        ascii_value <ax>
        pop     dx
        push    ax

        mov     al,dh
        mov     ah,0
        ascii_value <ax>
        push    ax

        ; display it
        mov     ax,0a000h
        mov     ds,ax
        pop     ax
        push    ax
        mov     ah,0
        mov     ds:[160*24],ax
        pop     ax
        xchg    ah,al
        mov     ah,0
        mov     ds:[160*24+2],ax

        pop     ax
        push    ax
        mov     ah,0
        mov     ds:[160*24+4],ax
        pop     ax
        xchg    ah,al
        mov     ah,0
        mov     ds:[160*24+6],ax

        ; register resume
        pop     dx
        pop     ds

        ; return value set
        mov     ah,0ffh
        iret

release_func    proc
        ; disable timer
        mov     ah,3
        push    ds
        pop     es
        mov     bx,offset my_work
        int     1ch

        ; clear time
        mov     ax,0a000h
        mov     es,ax
        mov     word ptr es:[160*24],0
        mov     word ptr es:[160*24+2],0
        mov     word ptr es:[160*24+4],0
        mov     word ptr es:[160*24+6],0
        ret
release_func    endp

intf_func       proc
        ; nothing
        ret
intf_func       endp

init_seg        segment 'stack'         ; for initialize only
        assume  cs:init_seg

check_func      proc    far
        ; check existense of timer resident program
        mov     ax,timer_id
        mov     bx,0
        mov     dx,bx
        int     2fh

        or      dx,dx
        @if     z
                ; not resident
                mov     dx,offset no_timer_mes
                @puts <dx>

                ; return with error
                mov     ax,1
        @else
                ; no error
                xor     ax,ax
        @endif
        ret
check_func      endp

init_func       proc    far
        ; interrupt regist
        ; work area initialize
        mov     my_work.timer_count,1
        mov     word ptr my_work.timer_routine,offset interrupt_func
        mov     word ptr my_work.timer_routine+2,cs

        ; time value set
        ; get dos time
        mov     ah,2ch
        int     21h

        mov     hour,ch
        mov     minute,cl
        mov     second,dh
        mov     tmsec,dl

        ; regist
        mov ah,5
        push    ds
        pop es
        mov bx,offset my_work
        int 1ch

        ; and resume
        ret

init_func       endp

init_seg        ends

        end
```


ワンラインNEWS
松下電器産業はパソコン分野で提携している富士通からICメモリカードを採用した16ビットノート型パソコンのOEM供給を受け品揃えを拡充した．

が，必要に応じてファイルを読みにいくので，別に記憶領域を必要としないという利点があります．

前者のプリンタスプーラを考える場合，大別して２つの機能をもつ必要があります．１つは，ファイルから自分のもっているバッファにデータを読み込んでくる機能であり，もう１つはバッファにたまったデータを，プリンタがあいているときに送出するという機能です．

ファイルの読み込み

DOSコールの利用

ファイルの読み込みにはDOSコールを利用する必要がありますから，これを実行する契機には十分に注意を払わないといけません．一般に，DOSに再入することは許容されません．これは，DOSがスタックなどの領域を固定的に使用しているため，再入すると前の状態の復元ができなくなるからです．このため，DOSを呼ぶ場合には，現在DOSを実行中でないことを保証してやる必要があります．

INT28の利用

これを実現するためには，いろいろなレベルでの対応のしかたがあります．まず，DOSではキーボードからの入力動作などを実行している最中に，INT28を起動します．DOSのファンクションはAH <=0BHのファンクションと，それ以上のファンクションに分けられています．このほかにもいくつかの特例はありますが，総じてこれら２つの組によって使用するワークエリアが分けられています．

このうち，INT28は低い番号のファンクションを実行中の場合にのみ呼び出されます．ですからINT28が呼び出されているときには，AH>0BHのファンクション，これらにはファイルアクセスなどが当然含まれますが，それをしてもよいことが保証されているのです． INT28でファイルアクセス中に致命的エラーが発生した場合などは，さらに別のワークエリアが使用されるので，とくに問題は生じません．逆に，致命的エラー発生中にAH<=0BHのファンクションを実行しても，INT28は発生しないようになっています．このように，INT28をトラップすれば，比較的容易にファイルアクセスなどに使用できるDOSのタイミングを把握できます．

この方法の難点は，当然のことながらINT28Hが発生する状況でしか有効でない，ということです．PC-9801では，IBM PCと違ってFEPの関係からキー入力はBIOSではなくDOSを使用するのか普通です．いろいろなアプリケーションで，入力待ち状態になっている場合，INT28は発生している場合が多いのです．

しかし，プログラムがなんらかの計算を実行している最中や，キーBIOS，キーワークエリアを直接使用してキー入力状態を把握しているようなプログラムの場合，必ずしもINT28は定期的には発生しません．また，CONをファイルハンドルでオープンして読み込みをしている最中にもINT28は発生しません．Cで普通にプログラムを書くと，このような状況になります．

仮にスプーラがファイルを読み込むのにこのINT28を使用したとすると，これらのようなプログラムを実行している最中では，ファイルの読み込みは行われないことになります．また，INT28をトラップするというやり方では，下位のファンクションは実行できないため，FEPを利用したキー入力を受け付ける，というようなことは実現できません．もし仮にキー入力が必要になる場合には，BIOSでも使用しないといけなくなります．

DOS／BIOSのトラップ

もうちょっと自由度と対応できる範囲を広げようとすると，DOSとBIOSのトラップをするということになります．BIOSはもちろん，DOSでも問題になるのは「行きっぱなし」のキーボードからの入力です．そのほかのファンクション，たとえば文字の出力やファイルへのアクセスなどは，比較的短時間の内に終了することが期待されます．ですから，DOSからもどってきた時点でトラップができれば，自由にDOSの機能を利用することができます．

これに対して，「入力があるまで待つ」というタイプの入力ルーチンだと，長い時間DOSに機能が渡りっぱなしになるので，入力が完了するまでの間DOSの呼び出しを実行することはできません．これは不都合なので，このようなファンクションは「展開」してやることにします．

たとえば１文字の入力というルーチンは，キー入力のステータスをチェックし，それがあった場合に１文字入力をするというループ構造で実現すると，DOSに滞留している時間は十分短くなります．その間に，いろいろな処理を行うことができるわけです．

キーBIOSについても同様です．BIOSでも行ったきりになるファンクションがありますので，これはステータスチェックと入力とに分解し，その間々でDOSを呼び出せるようにすればよいわけです．

このようにすると，キーボード入力

はもちろんのこと，プログラムがなんらかの形でDOSをアクセスしている場合には，トラップをかけることが可能となります．ただし，DOSをまったく呼ばないような状況でひたすら計算を実行しているようなプログラムの場合には，この方法でも対応することはできません．

タイマ割り込みの併用

DOSのアクセス可能性を最大限に保証しようとすると，さらにタイマ割り込みを併用する必要が出てきます．前記の計算を続けている状況では，タイマ割り込みでトラップをかけるようにするわけです．

ただし，タイマ割り込みでトラップがかけられるかどうか，ということを調べるのは，ちょっと厄介です．DOS呼び出しをトラップしておけば，DOSの中にいるのかどうか，ということを把握するのは容易そうですが，なかなかそううまくはいきません．ディスクアクセスの最中などに致命的エラーが発生すると，呼び出し元には帰らなくなります．そのままプログラムを終了させてしまうわけです．

この状況をトラップしているプログラムの側からみると，いつまでたってもDOSから戻ってこないように見えます．外から見ているだけでは，DOSの中にいるのかどうかを正確には判断できないのです．そのために，INDOSフラグというのが用意されています．このフラグはDOS自身が管理しているものですから，致命的エラーで中断した場合でも正常にDOSファンクションの終了を把握しています．これを参照することによって，DOSのファンクションが実行中であるかどうかを知ることができます．さらに厳密には，致命的エラーフラグも同時に参照し，どちらかがセットされていればDOSは使用できないと考えなければいけません．もちろん，タイマ割り込みがかかった時点でディスクBIOSなどDOSと共用しなければいけないBIOSなどの資源をアクセスしている場合にも，DOSを呼び出してはなりません．

これらの条件をまとめると，常駐プログラムでDOSをアクセスできるようにするいちばん有効な方法は，

①DOS／BIOSのエントリをトラップし，その呼び出し前後で必要に応じて処理をするようにする．
②DOS／BIOSの「行ったきり」になるファンクションは，それを分解してループに組みなおす．
③タイマ割り込みをかけ，INDOSフラグ，致命的エラーフラグその他各種BIOSなどを呼び出していないときに処理をするようにする．

というようなことになるでしょう．この手の話に完全な解決方法はなく，たとえばプログラムが直接ディスクやそのほかのチップをアクセスしている場合には，排他処理が完全とはいえず問題を起こすことがないとはいえません．

DOSのPRINTでは，DOSの起動チャンスのもっとも多い方法をとっているようです．しかし，実際にはINT28をトラップするだけでもかなり有効に働いてくれます．

データの送出

ポートを直接たたく

もう1つ要求される機能は，プリンタにデータを送り出すものです．これは，タイマで起動されるルーチンでプリンタのステータスチェックを行い，プリンタがあいている間にデータを送り込めばよいわけです．プリンタがBUSYになったら送出を中断して元のルーチンに戻らないといけないので，BIOSは使えずポートを直接たたくことになります．

この場合もハードウェアの競合のことを考えると，ほかのプリンタアクセスルーチンとの排他を考えないといけません．PRINTでは，INT5のコピールーチン，INT1AのプリンタBIOSルーチンなどをトラップしてチェックしています．もちろん，スプーラを起動している最中に表でハードウェアを直接たたくプログラムを実行した場合には，データが混ざる以上の問題が生じることになるでしょう．このあたりは，ユーザーの使いこなしで対処していかないとなりません．

タイマ割り込みの使用

以上のような機能を実現するためには，どうしてもタイマ割り込みを使用することが必要不可決です．PRINTはハイレゾモードのDOSの元で動いているときには，マルチイベントのタイマサービスを使用していますが，通常のDOSではINT8のハードウェア割り込みルーチンを直接乗っ取ってしまっています．このためせっかくtimerでマルチイベントのサポートをしたとしても，その機能がいかせなくなってしまいます．

ですから簡単なスプーラを作ってみようということになるわけです．

作成するスプーラの仕様

今回のスプーラは，簡単にするため

ワンラインNEWS
ライフボートはカナダWATCOM社のパソコン用プログラム言語「WATCOM COBOL」を日本語MS-DOSに移植し，COBOL学習用に改良して発売した．

にファイルの読み込みはINT28でのみ実行することとします．前記のような条件はありますが，INT28でもかなりの場合有効にファイルの読み込みを行うことができます．バッファにある程度空きができたら，そこにデータを読み込むという方法をとったほうが効率がよくなるのですが，今回はやはり簡単にするためにバッファ内のすべてのデータが出力されてしまってから，新しいデータを読みにいくこととします．

タイマ割り込みには当然timerのマルチイベントサービスを利用します．割り込みが起動されると，プリンタのステータスをチェックにいって，最高100バイトを1度に出力します．それ以前にプリンタがBUSY状態になれば，出力は打ち切ります．

スプーラは直接コードをポートに出力するということから，本来なら漢字変換を行わなければいけません．そうしないと，SJISの漢字コードがそのままグラフィックキャラクタとして出力されてしまいます．この漢字モードに移行するシーケンスはプリンタによって違うので，注意する必要があります．

今回は，ファイルのデータをそのままポートに出すようにしているので，漢字コードは正常に打ち出せません．そのかわり，グラフィックパターンなどは変換が実行されないので問題なく出力することができます．SJISコードをサポートする場合には，連載の最初のほうで画面アクセスルーチンをあつかっていたときに掲載したSJIS-JISの変換ルーチンを使用してプリンタに出力するようにすればよいでしょう．

プログラムの説明

spooler.asm

では，プログラムspooler. asmの内容を説明します．プログラムの最初のほうは，dispclkと同一です．このプログラムは，IDを2586Hとしています．また，ディスクバッファのサイズはbuffer_sizeで規定されていて，ここでは約8KBを取得しています．

INT28をファイルアクセスのためにトラップすることから，interrupt_listの内容は，INT28用の1つ分用意されています．データ領域は，やはりtimer用のワークエリアのほかに，ディスクバッファとそのポインタ，ファイルをアクセスするためのハンドルなどの変数領域が用意されています．メッセージは，dispclkのものと同一です．

int_28

int_28は，INT28のサポートルーチンで，必要ならファイルの読み込みを行うというものです．最初にDSレジスタの設定をした後，fdとポインタのチェックを行います．ファイルが出力されオープンされていない状況ではfdは-1とします．そうでなければ，プリント中のファイルがあるということを示しています．次に，バッファのポインタが，バッファのアクティブデータの最後尾に達しているかどうかを調べます．これらが一致すると，有効なデータは皆出力されたことになります．

以上の条件がともに満たされれば，バッファにデータの読み込みを行わないといけません．まずレジスタをセーブした後，PSPのすり替えを行います．

あるプロセスでオープンしたファイルハンドルは，そのプロセスのPSPが有効になっているときのみアクセスが可能です．これは，対象となるファイルをさす一種のポインタがPSPに記録されているからです．PSPの中の配列の添字をfdとした要素が，ファイル記述子（つまりはFCB）を指し示しているわけです．出力用のファイルはこのプログラムの中でオープンされますから，このプログラムのPSPをDOSに教えてやらないと，開いたファイルがアクセスできないのです．

PSPの取得はAH=51Hで，設定は50Hで行われます．PSPの受け渡しにはBXレジスタが使用されます．いままで使用されていたPSPの値は，ワークエリアに格納されます．それから，自前のPSPのもとでファイルの読み込みを行います．エラーがでなければ，ポインタ類を初期化します．バッファの終端は，読み込んだ長さに依存して設定されます．

エラーがおきた場合には，ステータスであるAXレジスタをクリアします．ファイル終端から読み込みを実行した場合には，エラーなし，AX=0の状態が返りますから，AXが0である場合には処理を中断してよい，という意味になります．

ファイルの送出終了，またはエラーが発生した場合には，まずファイルのハンドルをクローズします．それからfdの値を初期化します．その後に，PSPの値を復元します．そしてレジスタを復元し，INT28の元の割り込みルーチンにチェインします．

さて，このルーチンで問題になるのはPSPをすり替えている間です．PSPをすり替えてファイルアクセスにいったときに致命的エラーが発生すると，デフォルトの応答ではA／R／Iの入力を聞いてきます．RまたはIを選択した場合にはそのまま戻ってきますから，あまり問題はありません．しかしAを選択すると，PSPで示されるプログラムが終了してしまいます．ここでPSPがすり変わっているので本来処

ワンラインNEWS
シャープは最小・最軽量のノート型UNIXワークステーションを発売した．A4版より一回り大きいサイズで，従来の半分の小型化を実現した．

理されるべき表のプログラムが対象とならず，常駐プログラムのほうが終了されてしまいます．これはすぐにシステムのクラッシュを引き起こすでしょう．

これを回避するためには，自前のエラー処理ルーチンを用意する必要があります．エラーが発生した場合にはPSPを元に戻し，RやIで元のルーチンに復帰してきた場合にはPSPを常駐プログラムのそれに戻す，という処理をしなければいけません．今回は，エラーが生じた場合には決してAを実行しない，ということで我慢することにします．

interrupt_func

その次のinterrupt_funcはタイマ割り込みの処理ルーチンです．やはりデータセグメントをセットした後，ファイルがアクティブでかつバッファのポインタが末端に達していないで，プリンタが出力可能である場合にデータ出力を開始します．

SIレジスタにはバッファのポインタを格納し，1回に出力できる最大の文字数100文字はCXレジスタに設定します．

それから出力ループに入ります．ループではまずバッファからデータを取り出して，それをプリンタポートに出力した後STB信号にパルスを出します．前記のように，ここではデータの加工をまったく行っていません．単にファイルから読み込んだデータを出力するだけです．このため，SJISコードは漢字としては処理されません．これをサポートする場合には，ここにSJIS-JIS変換ルーチンを挿入することになります．また，もちろんモードを移すためのエスケープシーケンスも同様に出力する必要があります．

リスト4　spooler.asm

```
;------------------------------
; spooler: printer spooler
;
; $Log: RCS/timer.asm $
;
;------------------------------

        dosseg
        .model  small,pascal

        include condx.inc
        include utils.inc

        masm51

        .stack  2000h

; for interrupt
        .code
        public  my_id,interrupt_list,init_func,release_func
        public  check_func,intf_func

my_id   dw      2586h
timer_id        equ     2584h
buffer_size     equ     2000h

int_work        struc
int_num         dw      ?
int_func        dw      ?
int_old         dd      ?
int_work        ends

interrupt_list  label   byte
i28     int_work        <028h,int_28,0>
        dw      0

        .data
timer_work      struc
forward_link    dd      ?               ; forward link
timer_value     dw      ?               ; decrement count

timer_count     dw      ?               ; timer work initial value
timer_routine   dd      ?               ; execute routine on timeout

backward_link   dd      ?               ; backword link pointer
timer_status    db      ?               ; overflow flag, repeat mode
timer_divide    db      ?               ; timer divide work
timer_work      ends

my_work timer_work      <>

disk_buffer     db      buffer_size dup (?)
buffer_pointer  dw      disk_buffer
buffer_tail     dw      ?
fd      dw      -1
psp_work        dw      ?

no_timer_mes    db      'timer routine does not resident!',0dh,0ah,0

        .code

int_28:
        ; int28 trap routine
        ; DS set
        push    ds
        push    ax
        mov     ax,_DATA
        mov     ds,ax

        ; check buffer status
        @mc
                ; check file is active
                cmp     fd,-1
                @cond   nz

                ; check buffer empty
                mov     ax,buffer_pointer
                cmp     ax,buffer_tail
                @cond   z
        @if
                ; read buffer
                ; save registers
                push    bx
                push    cx
                push    dx

                ; change psp
                mov     ah,51h
                int     21h
                mov     psp_work,bx
                mov     bx,$psp
                mov     ah,50h
                int     21h

                ; device read
                mov     ah,3fh
                mov     bx,fd
                mov     cx,buffer_size
                mov     dx,offset disk_buffer
                int     21h

                ; pointer update
                cli
                @if     nc
                        ; read correctly
                        mov     buffer_pointer,offset disk buffer
                        mov     buffer_tail,offset disk_buffer
                        add     buffer_tail,ax
                @else
                        ; error, 0 bytes readed
                        mov     ax,0
                @endif
                sti

                or      ax,ax
                @if     z
                        ; file end or error
                        ; close file
                        mov     bx,fd
                        mov     ah,3eh
                        int     21h
                        mov     fd,-1
                @endif

                ; resume psp
                mov     bx,psp_work
                mov     ah,50h
                int     21h

                ; resume registers
                pop     dx
                pop     cx
                pop     bx
        @endif

        pop     ax
        pop     ds

        ; chain
        jmp     i28.int_old
```

```

interrupt_func:
        ; register save, DS set
        push    ds
        push    dx
        mov     ax,_DATA
        mov     ds,ax

        ; check condition
        @mc
                ; file is active
                cmp     fd,-1
                @cond   nz

                ; buffer is active
                mov     ax,buffer_pointer
                cmp     ax,buffer_tail
                @cond   nz

                ; printer is active
                in      al,42h
                test    al,4
                @cond   nz
        @if
                ; output 100char/buffer tail/not ready
                push    si
                push    cx

                ; set pointer/work
                mov     si,buffer_pointer
                mov     cx,100

                @begin
                        ; get buffer data
                        lodsb

                        ; put it
                        out     40h,al
                        jmp     $+2
                        jmp     $+2

                        ; strobe
                        mov     al,0eh
                        out     46h,al
                        jmp     $+2
                        jmp     $+2
                        mov     al,0fh
                        out     46h,al

                        ; pointer update
                        mov     buffer_pointer,si

                        ; waste time
                        jmp     $+2
                        jmp     $+2
                        jmp     $+2
                        jmp     $+2

                        ; check terminate condition
                        @mc
                                ; buffer empty
                                cmp     si,buffer_tail
                                @cond   z
                        @or
                                ; chars
                                dec     cx
                                @cond   z
                        @or
                                ; printer status
                                in      al,42h
                                test    al,4
                                @cond   z
                        @or
                        @if
                                @break
                        @endif
                @again

                ; resume registers
                pop     cx
                pop     si
        @endif

        pop     dx
        pop     ds

        mov     ah,0ffh
        iret

release_func    proc
        ; disable timer
        mov     ah,3
        push    ds
        pop     es
        mov     bx,offset my_work
        int     1ch
        ret
release_func    endp

intf_func       proc
        dec     bx
        @if     z
                ; bx=2, check status
                cmp     fd,-1
                @if     z
                        ; ready
                        mov     dx,0
                @else
                        ; busy
                        mov     dx,-1
                @endif
        @else
        dec     bx
        @e_if   z
                ; bx=3, assign file
                cmp     fd,-1
                @if     nz
                        ; busy, error
                        mov     dx,-1
                @else
                        ; change psp
                        mov     ah,51h
                        int     21h
                        mov     psp_work,bx
                        mov     bx,$psp
                        mov     ah,50h
                        int     21h

                        ; try to open
                        push    ds
                        push    es
                        pop     ds
                        mov     ah,3dh
                        mov     al,0
                        int     21h
                        pop     ds

                        ; check status
                        @if     nc
                                ; open ok
                                mov     fd,ax
                                mov ax,offset disk_buffer
                                mov     buffer_pointer,ax
                                mov     buffer_tail,ax
                                mov     dx,0
                        @else
                                mov     dx,-1
                        @endif

                        ; resume psp
                        mov     bx,psp_work
                        mov     ah,50h
                        int     21h
                @endif
        @endif
        ret
intf_func       endp

init_seg        segment 'stack'         ; for initialize only
        assume  cs:init_seg

check_func      proc    far
        ; check existense of timer resident program
        mov     ax,timer_id
        mov     bx,0
        mov     dx,bx
        int     2fh

        or      dx,dx
        @if     z
                ; not resident
                mov     dx,offset no_timer_mes
                @puts <dx>

                ; return with error
                mov     ax,1
        @else
                ; no error
                xor     ax,ax
        @endif
        ret
check_func      endp

init_func       proc    far
        ; interrupt regist
        ; work area initialize
        mov     my_work.timer_count,1
        mov     word ptr my_work.timer_routine,offset interrupt_func
        mov     word ptr my_work.timer_routine+2,cs

        ; get psp
        push    ds
        mov     cx,_TEXT
        mov     ds,cx
        assume  ds:_TEXT
        mov     ax,my_id
        pop     ds
        assume  ds:_DATA

        mov     bx,0
        int     2fh

        ; save psp
        mov     $psp,dx

        ; regist
        mov ah,5
        push    ds
        pop es
        mov bx,offset my_work
        int 1ch

        ; and resume
        ret
init_func       endp

init_seg        ends

        end
```

それからポインタを更新し，多少時間をつぶします．これは，正常にデータが受け付けられた場合にBUSYでなくなるまで待つためのものです．

ループの終了条件，まずポインタがバッファ端に達した場合，100文字出力が終了した場合，プリンタがBUSYになった場合にループを抜けます．そうでなければ，次の文字出力に取り掛かります．　最後に，レジスタを復元し，AH=0FFHの継続コードを返します．

release_func

release_funcは，dispclkと同様にタイマサービスのワークエリアを解放するためのものです．intf_funcには実体が定義されています．拡張サービスとして，BX=2でプリント状態の取得，BX=3でプリントファイルの指定とします．

まずBX=2であれば，fdをチェックし，これが-1であるとDXに0を設定します．これはプリントの受け付け可能を示します（PRINTのように出力ファイルキューはもっていません）．そうでなければ非0の値を返します．

次にBX=3であれば，やはりfdをチェックし，受け付け可能な場合にファイルをオープンします．まず，ファイル読み込みのときと同様にPSPのすり替えを行い，ES：DXで示されるアドレスのファイル名のファイルを開くことにします．この場合も，必要なら致命的エラールーチンをトラップすることになります．

ファイルのオープンが正常に受け付けられれば，fdとそのほかのポインタに値を設定し，DXに0を格納します．オープンエラーであれば，ステータスBUSYの場合と同様にDXには非0を返します．最後にPSPを復元して復帰します．

check_funcの内容はdispclkと同一です．init_funcでは，タイマカウンタ初期化のかわりに，PSPの設定手続きをしています．＄pspはutils.incの中で定義されていますが，localな変数であり，しかもそれに値が設定されるのは，スタートアップルーチンが使用されているモジュールでしか使用できません．ですから，このモジュールでプログラムのpspの値を知る必要がある場合には，あらかじめローカルな変数に値をコピーしておくことにします．

異なるモジュールにある同一の名前の変数をアクセスするのは不可能なので，ここでは自分のINT2Fサービスルーチンに対して常駐チェックのコマンドを発行することでPSPの値を得ています．

プログラムの作成方法

spoolset.c

プログラムの作成方法はdispclkの場合と同様です．モジュールを別々にアセンブルし，リンクしてプログラムを構成します．このプログラムを常駐させるだけではなにも起こりません．スプーリングを実行するためには，出力するファイルを指定しなければいけません．

そのためのプログラムをspoolset.cに示します．TCの疑似レジスタ変数を利用する非常に簡単なプログラムです．まず常駐チェックをした後，引数がなにもなければプリント状態取得のコマンドを発行し，状態を出力します．

引数があれば，それをファイル名と解釈してアドレスをES：DXに格納し，ファイルセットのコマンドを発行します．そして返り値によって適切なメッセージを出力します．

今回のプリンタスプーラは，必要最低限の機能をもっているだけのものです．実用に供するには漢字対応が不可欠でしょうし，プリントキューももてるようにしたほうがよいでしょう．場合によっては，ファイル読み込みの契機を拡大する必要が出てくるかもしれません．しかし，存在しているプログラムに機能を付け加えていくというのは比較的容易なものです．常駐スケルトンも，少しずつ手を加えながら随分使い回しています．これと同じように，形が1つあるとかなり楽だと思われます．

漢字対応などの機能は，練習のためにぜひ読者に付け加えていただきたいと思います．

なお，最近のTASMでは，condx.incの中の$xlist／$listが正常にアセンブルできないようです．これを避けるためには，condx.incの中のdebug指定を1にするか，あるいはこれらを文字マクロではなく通常のマクロ命令として定義すればよいようです．

リスト5　spoolset.c（Turbo C++対応）

```
#include <stdio.h>
#include <dos.h>

#define SPOOL_ID 0x2586

main(int argc, char **argv)
{
	_AX = SPOOL_ID;
	_BX = 0;
	_DX = 0;
	geninterrupt(0x2f);

	if (_DX) {
		printf("spool does not exist¥n");
		exit(1);
	}

	if (argc == 1) {
		/* check status */
		_AX = SPOOL_ID;
		_BX = 2;
		geninterrupt(0x2f);

		if (_DX)
			printf("printing¥n");
		else
			printf("ready¥n");
	}
	else {
		/* set file */
		_ES = _DS;
		_DX = (unsigned)argv[1];
		_AX = SPOOL_ID;
		_BX = 3;
		geninterrupt(0x2f);

		if (_DX)
			printf("error¥n");
		else
			printf("spool ok¥n");
	}
}
```


ワンラインNEWS
旭エレクトロニクスは米センス8社が開発したバーチャルリアリティシステム「ワールド・ツール・キット」を輸入・販売する．IBM-PCまたはPC互換機上で稼働

情報処理技術者試験

第2種合格！トレーニング

第9回 FORTRAN

富士情報ビジネス専門学校　稀代智明

FORTRANの出題傾向

FORTRANの問題に限らず，最近のプログラム言語に関する出題は，構造化されたプログラムを意識して作られていて，アルゴリズムの読解力が試されている．逆に高度な文法的知識を求める問題はほとんど出題されないので，基本的な文法を修得したら，読解力をつけるために，整列，表探索などの代表的なアルゴリズムに目を通し，また過去に出題されたプログラムを熟読し，完全に理解しておくことが必要である．

過去の問題を分析すると，数多く出題されているのが，ソート(整列)の問題，基本的な統計問題とグラフの作成である．統計についての基本的な考え方(平均，分散，標準偏差，相関係数など)とその計算の仕方は知っておいて損はない．またグラフの作成に関する問題も慣れることが大切なので，多くの問題にあたってみることが必要であろう．

文法的に高い頻度で出題されているのは，DO文，ブロックIF文，配列，文字列操作，主プログラムと副プログラムについてである．これらの文法の基本はまとめておく必要があるし，型宣言文や書式仕様における編集記述子，基本的な組み込み関数の知識は必修である．

●表1　FORTRANの出題内容(テーマ)

出題内容	88		89		90		91
	春	秋	春	秋	春	秋	春
統計計算	⑥⑩	⑭	⑭	⑭		⑩	⑥
文字列処理	⑭						
基本算法				⑥	⑥		
グラフ作成		⑥	⑭		⑩	⑩	⑭
整列				⑩	⑭	⑭	⑭
その他			⑥⑩				⑩

数字は問題番号を示す

ソート（整列）

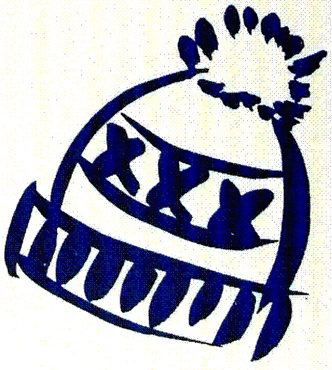
基本的なアルゴリズムの中で，よく登場するのが整列のアルゴリズムである．整列の基本形には，挿入法，交換法，選択法の3つがあるが，例題では交換法の代表的なものとして隣接交換法を取り上げる．

●出力例1

```
185          ｷﾑﾗｱﾂｼ
183          ﾖｺﾀﾄｼﾉﾘ
180          ｱｲｻﾞﾜﾋﾛｼ
178          ｱﾍﾞｱｷｵ
175          ｽｽﾞｷｻﾄｼ
172          ｻﾄｳｶﾂﾋﾛ
169          ｵｷﾀｱｷｵ
168          ﾀﾅｶｻﾄﾙ
166          ｲｲﾀﾞｶｽﾞｼｹﾞ
162          ﾅｶﾔﾏﾏｻｼ
```

EXAMPLE1

カードから身長と氏名を読み込み，身長の高い順(降順)に整列し身長，氏名を印字するプログラムを作成しなさい．ただし，入力するデータ件数は最大200件とし，入力は標準入力形式で行うものとする．



●リスト１

```
 1:       CHARACTER*20 NAME(200),SNAME
 2:       INTEGER SINTYO(200),SSINTYO,NIN,I,J,M
 3:       DO 10 NIN=1,200
 4:          READ(5,*,END=100) NAME(NIN),SINTYO(NIN)
 5:    10 CONTINUE
 6: C
 7:   100 NIN=NIN-1
 8:       DO 30 I=NIN-1,1,-1
 9:          DO 20 J=1,I
10:             IF(SINTYO(J).LT.SINTYO(J+1)) THEN
11:               SSINTYO=SINTYO(J)
12:               SINTYO(J)=SINTYO(J+1)
13:               SINTYO(J+1)=SSINTYO
14: C
15:               SNAME=NAME(J)
16:               NAME(J)=NAME(J+1)
17:               NAME(J+1)=SNAME
18:             ENDIF
19:    20       CONTINUE
20:    30    CONTINUE
21: C
22:       WRITE(6,1000) (SINTYO(I),NAME(I),I=1,NIN)
23:  1000 FORMAT(' ',I5,10X,A)
24:       END
25:
```

解　説

リスト１は解答例である．

1～2行目は変数，配列の型宣言，同時に配列の要素数も宣言している．3～5行目ではDOループにより配列へ入力データをセットしている．END指定は200件データがない場合のためである．７行目は，入力データの件数(＝人数)を調整するためのもの．FORTRANのDO文はwhile型の繰り返し制御を行うため，ループの終了時DO変数NINは入力データより１つ多くなる．そのため７行目でNINから１を引いてやることでデータ件数＝NINとなるように調整している．

8～20行目は隣接交換法によるソートの本体である．この方法の基本は配列の先頭から順番にとなりあったデータを比較し，降順(または昇順)の関係になっていないならば，それを交換することを，すべてソートが終わるまで繰り返すことである．このプログラムの場合，SINTYO(J)<SINTYO(J+1)ならばSINTYOとNAMEの(J)と(J+1)の要素をそれぞれ交換し，J＝Iとなるまでそれを行うと，SINTYOとNAMEの(I+1)の位置に全体で最も小さい身長を持つデータがセットされる．後はソートの範囲Iを１ずつ減らして同様に繰り返し，I＝1になったとき，ソートが完了することになる．

10行目の条件式をSINTYO(J)>SINTYO(J+1)とすると昇順にソートされる．

配列を使用した場合必ずDO文による繰り返し処理が登場すると考えてよい．その解析は添字を表すDO文の制御変数のふるまいに注意すればそれほど困難ではない．

グラフの作成

グラフを作成する問題も数多く出題されている．基本的な統計処理と組み合わされての出題も多いが，グラフの作成ということに限ると，文字型変数の扱いと部分文字列の指定に注意しなければならない．例題として，簡単なグラフ出力プログラムを取り上げる．

EXAMPLE2

ある地点における各月の雨量データを入力し，横ヒストグラムを出力する．各月の最大雨量を300mmとし10mm毎に'＊'を印字する．ただし10mm未

●表２　比較演算子と論理演算子

比較演算子	意味	論理演算子	意味	優先順位
.GT.	$>$	.NOT.	否定	高
.GE.	$\geqq$	.AND.	論理積	↓
.EQ.	$=$	.OR.	論理和	↓
.NE.	$\neq$	.EQV.	論理等価	↓
.LT.	$<$	.NEQV.	論理非等価	低
.LE.	$\leqq$			

●出力例 2

```
        0        100       200       300(mm)
        +---------+---------+---------+
    1   I***
    2   I*******
    3   I**********
    4   I************
    5   I**************
    6   I******************
    7   I***************
    8   I****************
    9   I*************************
   10   I************************
   11   I*********
   12   I*****
        +---------+---------+---------+
        0        100       200       300(mm)
```

●リスト 2

```
 1:       CHARACTER*30 STAR,GRAPH
 2:       INTEGER URYO,I,N
 3:       DATA STAR/'******************************'/
 4:       WRITE(6,2000)
 5:       WRITE(6,2010)
 6:  2000 FORMAT(' ',4X,'0        100       200       300(mm)')
 7:  2010 FORMAT(' ',4X,3('+---------'),'+')
 8: C
 9:       DO 10 I=1,12
10:          READ(5,*) URYO
11:          N=URYO/10
12:          GRAPH=STAR(1:N)
13:          WRITE(6,1000) I,GRAPH
14:  1000    FORMAT(' ',I2,2X,'I',A)
15:    10    CONTINUE
16:       WRITE(6,2010)
17:       WRITE(6,2000)
18:       END
19:
20:
```

満は切り捨てとし'＊'は印字しない。

解　説

リスト2は解答例である。

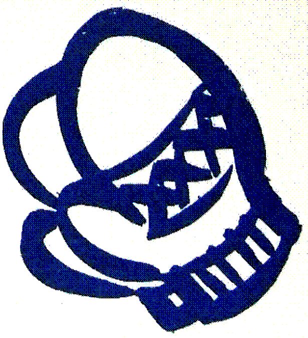
3行目はDATA文による初期値の設定で，文字型変数STARTに30個の'＊'を設定している。4，5行目はグラフの目盛り印字を行う。

9～15行目でこのプログラムのメインであるヒストグラムの出力を行う。制御変数Iは月を表していて，I＝1のとき1月のグラフを，I＝2のとき2月のグラフをというようにそれぞれ印字し，I＝12まで繰り返す。

11行目の計算式は'＊'1個が10mmの雨量であるから，雨量を10で割り，雨量に相当する'＊'の個数をNに求めている。ちなみに，この計算は整数型で行われるため，10で割ったときの商の小数点以下は切り捨てられる。結果的に10mm以下の雨量は切り捨てとなり，問題の条件を満たすことになる。

12行目は文字型変数GRAPHにNに相当する個数の'＊'をセットする。その際，部分文字列指定を用いて，あらかじめ30個の'＊'がセットされた文字型変数STARTから，必要な個数の'＊'を切り取りGRAPHに送っている。

13行目でI（月）とGRAPHを出力している。



EXERCISE

EXAMPLE2と同じ条件で縦ヒストグラムを出力するプログラムを作成した．空白をうめてプログラムを完成しなさい．

●出力例 3

```
 300 +            +
     I            I
     I            I
     I            I
     I            I
     I        *   I
     I        **  I
     I        **  I
     I        **  I
     I        **  I
 200 +        **  +
     I        **  I
     I     *  **  I
     I     *  **  I
     I     * ***  I
     I     *****  I
     I    ******  I
     I    ******  I
     I   *******  I
     I   *******  I
 100 +  ********  I
     I  ********* I
     I  ********* I
     I ********** I
     I ********** I
     I ***********I
     I ***********I
     I************I
     I************I
     I************I
   0 +------------+
               111
      123456789012
```

●リスト 3

```
 1:       CHARACTER GRAPH(1:30)*12
 2:       INTEGER URYO(12),I,J,N
 3:       DO 10 I=1,12
 4:          READ(5,*) URYO(I)
 5:    10    CONTINUE
 6: C
 7:       DO 30 I=1,12
 8:          N=[   (1)   ]
 9:          DO 20 J=1,N
10:             [   (2)   ]
11:    20       CONTINUE
12:    30    CONTINUE
13: C
14:       DO 40 J=[   (3)   ]
15:          IF(MOD(J,10).EQ.0) THEN
16:             WRITE(6,100) J*10,GRAPH(J)
17:   100       FORMAT(1X,I3,' +',A12,'+')
18:          ELSE
19:             WRITE(6,110) GRAPH(J)
20:   110       FORMAT(1X,'    I',A12,'I')
21:          ENFIF
22:    40    CONTINUE
23: C
24:       WRITE(6,200)
25:   200 FORMAT(3X,'0 +',12('-'),'+')
26:       WRITE(6,210)
27:   210 FORMAT(15X,'111')
28:       WRITE(6,220)
29:   220 FORMAT(6X,'123456789012')
30:       END
31:
32:
```

解　説

縦ヒストグラムの場合，最初に配列の中にグラフを作成しておいて，最後に出力するやり方が一般的であり，このプログラムでもそれにしたがっている．

7～12行目で配列GRAPHに'*'をセットし，グラフを作成している．GRAPH(1)には10mm以上の雨量，GRAPH(2)には20mm以上の雨量があったときそれぞれ'*'がセットされ，各月の雨量を'*'の個数に換算した値Nの位置まで，つまりGRAPH(N)までそれを行う．その時，配列GRAPHの部分文字列(1:1)～(12:12)がそれぞれ1月から12月に相当しているので，たとえば12月に50mmの雨量があったとすると，GRAPH(1)～GRAPH(5)の(12:12)の位置に'*'がセットされればよい．

14～22行目でグラフを出力する．GRAPH(30)からGRAPH(1)の順に出力すればよい．

解　答

(1)　URYO(I)/10

(2)　GRAPH(J)(I:I)

(3)　30,　1,　-1

統計処理

統計処理プログラムの問題は，統計に関する知識がなくても，問題文をよく読めば解答するのは決して難しくはない．しかし，平均，分散，標準偏差などの算出方法は知識として必要であろう．

例題では簡単な統計処理を取り上げる．

EXAMPLE3

英語，数学，国語の3教科の試験結果を読み込んで，個人総得点，各教科の平均点と分散，標準偏差および，総得点の平均点と分散，標準偏差を計算

●出力例 4

```
911001    82    75    91     248
911003    64    81    38     230
911004    83    92    52     227
911005    60    64    83     207
911007    71    60    65     196
911008    46    40    37     123
911009    94    99   100     293
911010    85    71    76     232
911012    48    60    49     157
911015    58    38    44     140
AVE.    69.1  68.0  63.5   200.6
VAR.   248.9 384.1 508.6  2549.8
STD.    15.8  19.6  22.6    50.5
```

●リスト 4

```
 1:       INTEGER NO(200),TEN(200,4),KEI(4),N,I,J
 2:       REAL HEIKIN(4),BUNSAN(4),HENSA(4)
 3: C
 4:       READ(5,*) N
 5:       DO 10 J=1,4
 6:          KEI(J)=0
 7:    10    CONTINUE
 8: C
 9:       DO 30 I=1,N
10:          READ(5,*) NO(I),(TEN(I,J),J=1,3)
11:          TEN(I,4)=TEN(I,1)+TEN(I,2)+TEN(I,3)
12:          DO 20 J=1,4
13:             KEI(J)=KEI(J)+TEN(I,J)
14:    20       CONTINUE
15:    30    CONTINUE
16: C
17:       DO 50 J=1,4
18:          HEIKIN(J)=REAL(KEI(J))/REAL(N)
19:          BUNSAN(J)=0.0
20:          DO 40 I=1,N
21:             BUNSAN(J)=BUNSAN(J)+(REAL(TEN(I,J))-HEIKIN(J))**2
22:    40       CONTINUE
23:          BUNSAN(J)=BUNSAN(J)/REAL(N)
24:          HENSA(J)=SQRT(BUNSAN(J))
25:    50    CONTINUE
26: C
27:       DO 60 I=1,N
28:          WRITE(6,200) NO(I),(TEN(I,J),J=1,4)
29:    60    CONTINUE
30:       WRITE(6,210) (HEIKIN(J),J=1,4)
31:       WRITE(6,220) (BUNSAN(J),J=1,4)
32:       WRITE(6,230) (HENSA(J),J=1,4)
33:   200 FORMAT(' ',4I6,I8)
34:   210 FORMAT(' AVR.   ',3F6.1,F8.1)
35:   220 FORMAT(' VAR.   ',3F6.1,F8.1)
36:   230 FORMAT(' STD.   ',3F6.1,F8.1)
37:       END
38:
39:
```

し，出力するプログラムを作成しなさい．ただし，件数は200件を越えないものとする．

解　説

リスト4は解答例である．

5～7行目は合計領域KEIの各要素をゼロクリアしている．KEI(1)～KEI(4)にはそれぞれ英語，数学，国語，総得点の合計が求められる．

●平均・分散・標準偏差の算出方法

平均　$\bar{x}=\frac{1}{n}\sum_{k=1}^{n}a_k$

分散　$\sigma^2=\frac{1}{n}\sum_{k=1}^{n}(a_k-\bar{x})^2$

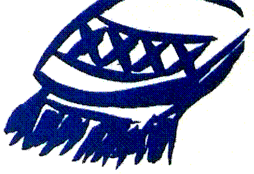
標準偏差　$\sigma=\sqrt{\sigma^2}$

9～15行目はデータを1件ずつ読み込んで，総得点の計算と各合計の累計を行う処理を人数分繰り返している．

10行目は入力のDO形並びが使われているが，配列に対する入力にはよく用いられる方法である．

17～25行目は統計処理のメインで，平均，分散，標準偏差を算出している．この中で文法的に重要な事項は，18，21，23行目の算術計算において型を合わせるところである．計算結果を代入する配列の要素が実数型なので，計算式内で，整数型の変数または配列要素について，REAL関数により実数型に変換している．算術式および算術代入文における型の統一は，注意を怠ると予想外の結果が得られる元ともなるので，注意が必要である．また，ここで使われているREAL関数およびSQRT関数（平方根を求める）以外にも，組み込み関数が用意されているので，基本的な関数について，その意味，引数の数，引数の型，結果の型について暗記しておいたほうがよい．

最後に27～32行目で結果を出力している．ここで使用されている出力のDO形並びは，前述の入力のDO形並びと同様，配列の出力時によく用いられる．

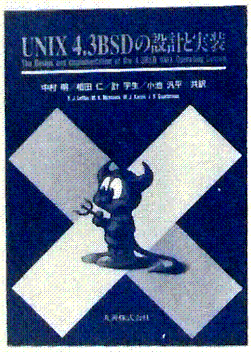


Book Review

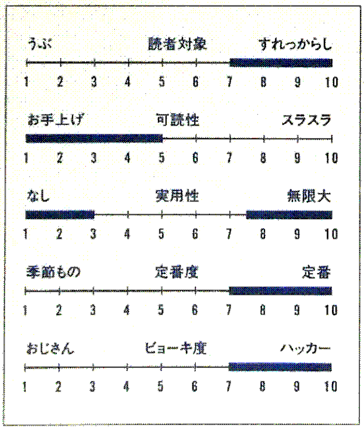
■中村正三郎

■UNIX4.3BSDの設計と実装

S.J.Leffler, M.K.Mckusick, M.J.Karels, J.S.Quarterman著
中村明, 相田仁, 計宇生, 小池汎平訳
丸善　5150円



■評価チャート



設計開発者たち自らが技術的到達点を伝える

ハッカーが使うUNIXは，BSD(Berkeley版)でなければならない．かつてそういう時期があった．AT&TのSystemVはおよびではなかったのだ．BSDのバージョンでいえば，4.1BSDが登場してから，本書が対象としている4.3BSDの時代までに相当する．現在のUNIXの主流はSun OSに移っているが，これもBerkeley版がベースになっている．

なぜUNIXなのか．しかもなぜBSDなのか．それは，BSD UNIXがさまざまなアイデアをぶち込んだ，コンピュータサイエンスの最先端の実験場だったからだ．実用面においても，仮想記憶のサポート，ソケットによるプロセス間通信，TCP/IPによるネットワークのサポートなど，先進的で魅力にあふれる機能が実装されていた．BSDは，UNIXの歴史における中興の祖といってよく，中でも4.3BSDは，ひとつの頂点を極めたバージョンである．

本書は，4.3BSDの設計開発者たち自らが，その内部構造を詳述したものである．全体を，カーネルサービス，プロセス管理，メモリ管理，入出力システム（ファイルシステム，デバイス管理），ネットワーク(プロセス間通信，プロトコル）に分け，それぞれの実装上のアイデア，データ構造，アルゴリズムを詳しく提示している．

といって無味乾燥な仕様書ではなく，随所に設計上の判断に苦しんだことをうかがわせる記述がある．OSの設計開発者はもとより，趣味的にOSに興味を持つ者にとっても，得られる示唆は大きい．また本書はOSの教科書としての側面も持つ．何よりUNIXは，OSの生きた教材として親しまれてきたのだ．4.3BSDは，UNIX文化のひとつの技術的成熟に到達したバージョンである．その詳しい解説書が，OSの教科書としても十分な魅力を持つのは当然だろう．

原書は2年前に出版されて話題になった．進歩の早い世界で，このタイムラグはありすぎる．鮮度が落ちているのは否めない．その間，世の中の主力UNIXは，BSDからSun OSに移り，あるいはMachへ移ろうとしている．もっと早い出版が望まれた．しかしながら，本書の内容が役に立たない時代遅れと思うのは早計だ．事実は逆で，4.3BSDに実装してあるものは，現在でもUNIXの中核技術と呼べるものである．子孫であるSun OSを知る上でも重要だ．さらにいえば，Machは，4.3BSDのカーネルへのアンチテーゼになっている．4.3BSDのカーネルは，とくにネットワークサポートによる影響から，複雑化，肥大化した．Machは，マイクロカーネルとメッセージパッシングのモデルによって，すっきりとしたOSを構築している，その上に4.3BSDを再構築しているので，BSDのソフトが動く．Sun OS, Machどちらも4.3BSDを抜きに語れない．巻末の用語集も充実している．訳はこなれていないが，UNIXハッカーなら，とにかく目を通しておきたい．

本書以外なら，Prentice Hallから出ているシリーズがUNIX系OSの中身を知るのに都合がよい．ComerのXINU，TanenbaumのMINIXはUNIX流OSを作ることによってOSの中身を明らかにしたものである．こ

ワンラインNEWS
東京エレクトロンは米コンベックスコンピュータ社と共同開発した普及型スーパーコンピュータを発売した．性能をアップしながら価格は従来機と同等に押さえている．

こでは邦訳があるものを紹介する.

本家AT&TのSystem V系の内部を知りたいのなら，Maurice J. Bach著，坂本文，多田好克，村井純訳「UNIX カーネルの設計」(共立出版)がたいへんに詳しい.

メッセージパッシングモデルによるモダンな設計で，UNIX Version 7互換のOSを再構築してみせ，OSの教育にも適しているのがMINIXだ. Andrew S. Tanenbaum著，坂本文監修，大西照代訳「MINIX オペレーティング・システム」は，OSの教科書としても優れている. MINIXは現在でも進化を続けている. 最新のMINIXの情報は，日経MIXなどで入手できる.

洋書の定番もの，和書では訳本が健闘！

古田島義和（神田・書泉グランデ）

コンピュータの方向が分散処理へと向かっている今，分散処理システム関係の図書が動きはじめています. しかし良好図書が少なく，和書では電気書院の「分散システム―コンセプトとデザイン」が91年夏に出版されて以来，順調な売れ行きを見せています. 洋書ではAddison Wesley（以下AW）の「Distributed System」（Mullender著）が最近になって売れはじめてきました. Prentice Hall（以下PH）の「Principles of Distributed Database System」（OZSU著）が堅実に売れているほか，Springer Verlagの「Art of Distributed Applications」（Carbin著）が最近入荷して話題を呼んでいます.

話題といえば「Macintosh Way」（Macintoshの開発に参加した人の秘話や苦労話を中心に書かれた本で，小社でも洋書がかなり売れました）の著者でもあるGuy Kawasaki氏がPeachpit Pressから「Database 101」（ディスク付）を出版し，入荷と同時に売れ行き好調です.

洋書のベスト10を見るとオブジェクト指向が上位を占めています. そしてMacintosh，Mathematica，MS-DOSと，その分野の定番的存在ともいえる図書が集中しています. ちなみに11位はAWの「Compilers」，12位はM. I. T Pressの「Structure and Interpretation of Computer Programs」（訳本はマグロウヒルの「プログラムの構造と実行」上・下巻），13位はMicrosoft Pressの「Programming Windows 2/E」です. 本書の訳本が10月にアスキーから「プログラミングWindows Version3」というタイトルで出版され，数週間前に出版された富士ソフトウェアの「Windows 3.0パワープログラミング」（上・下巻）以上に動きがよく，著者Petzold氏の人気の高さがわかります. 以上のように，11位から13位までも定番といえる図書が占めており，訳本が出版された時はベストセラーまちがいなしといえるでしょう. その証として和書のベスト10の半分が訳本で，洋書もベストセラーにあげられていました.

先月紹介しましたWindows関連の新刊が数点ありますのでご紹介しましょう. PHからは「Keys to Windows Utilities and Accessories」，「Keys to Microsoft Window 3. X」，「Keys to Wordperfect for Windows」（すべてVer. 3. 1対応），年末注目を集めるVisual BASICの関連書は，すでにAW，PH，Sybex，MIS，Bantamなど，いろいろな出版社から続々と入荷の予定です. そして期待の1冊として，92年2月ころAWから出版予定の「Undocumented Windows」があります. これは，A. Schulman Programming Seriesの第1弾で，今後のタイトルが楽しみです.

最後にNeXTの話題. 関西の大学でNeXTを数百台も購入したという話を聞いてビックリしましたが，そこで紹介したいシリーズ. AWの「NeXT Developer's Library」（全5巻），「Programming the Display PostScript System with NeXTStep」が今年暮れから来年1月か2月ころに入荷する予定ですので，NeXTユーザーのみなさんお楽しみに…….

■和書&洋書ベスト10

（書泉グランデ調べ：1991年9月1日～9月30日）

1	コンピュータ・ネットワーク	丸善
2	近代プログラマの夕	アスキー
3	UNIX Cプログラミング	アスキー
4	電脳騒乱節Vol.3	技術評論社
5	C98スーパーライブラリ	ソフトバンク
6	UNIXシステムチューニング	アスキー
7	WINDOWS3.0パワープログラミング上	富士ソフトウェア
8	Macintosh Bible Third Edition日本語版	技術評論社
9	WINDOWS 3.0パワープログラミング下	富士ソフトウェア
10	電子立国　日本の自叙伝　上	日本放送出版協会

1	C++ Programming Language 2/E	Addison Wesley
2	Object-Oriented Modeling and Design	Prentice Hall
3	ResEdit Complete	Addison Wesley
4	C++ Primer 2/E	Addison Wesley
5	Exploring Mathematics with Mathematica	Addison Wesley
6	Design & Implementation of 4.3 BSD UNIX Operating System Answer Book	Addison Wesley
7	Inside Macintosh V6	Addison Wesley
7	MS MS-DOS Programmer's Reference	Microsoft Press
7	Programming Perl	O'Reilly
10	Managing NFS & MIS	O'Reilly

SE-NET拾い読み

本誌掲載プログラムのダウンロードサービス用としてスタートしたSE-NETも開局以来ほぼ1年を迎え，ホストマシンとホストプログラムのバージョンアップが行われました．10月には常連を中心としたオフラインミーティングも開かれ，SE-NETもプログラマーズ・ページのもう1つのページとして少しずつ成長してきているようです．
ここでは，SE-NETの「プログラマの内輪話」ボードからいくつかの楽しい書き込みをご紹介します．また，書き込みについては編集の都合から一部を修正・割愛させていただきました．

「プログラマの内輪話」ボード

■サポートは……という話

ＲＲＲＲ・・・ＲＲＲ／Axonohydroria
平成03年08月19日（月）　午後06時09分25秒

「はい、サポートです。」
「お宅のソフトが動かないんですが。」
「どのような状態ですか。」
「同じところを何度も繰り返すだけなんです。」
「なるほど、画面は何が表示されますか。」
「えーと、スイッチを入れるとしばらくしてMS-DOSなんとかというのが出て、それからプログラムの名前みたいのが出ます。その後『何かキーを押してください』という表示が出ます。そこでキーを押すと画面が消えてまたMS-DOSなんとかへ戻ってしまい繰り返すだけなんです。」
「えっ！な… 何のキーを押したんですか？」
「は？えっと、お宅の人が来たときに『何かあったらこれを押してください』といっていたので、解り易いし、いいと思ったんで本体に付いているRESETキーを押したんですが。」
「・・・・・・」
実話とはいえあまりにも恐ろしい。私は大爆笑しましたが（もちろん影で）

Axonohydroria

某月某日のエピソード／Meg-8823
平成03年08月20日（火）　午後07時20分03秒

「ええっとですね，CONFIG.SYSの中のFILES＝を20位にしてみてください」
…
「エディタかなにか．．あ，ワープロはちょっと．．」
…
「エディタ… 何使ってますか？　は？　エドリン？」
サポート中の所長さんと顔を見合わせてしまった…

EDLINかぁ．．
もうコマンドをすっかり
忘れた★よつもと

「何か」というキー……／キタキツネ
平成03年08月22日（木）　午前08時35分22秒

うーん、私もまだパソコンをいじり始めてからそんなにたっている訳ではないので偉そうなことは言えないのですが。
でも、笑っちゃいけませんよね。本人は真面目なんですから。
そういえば、何かキーを押してくださいというメッセージに対して『「何か」というキーはいったい何処にあるのですか』と聞いてきた人もいたとか。

北きつね

■EDLINの教祖様？

会社の研修でEDLINしかなかった……／トラトラトラ
平成03年08月21日（水）　午前02時44分25秒

前の会社の研修でEDLINしかなかったのを思いだした。
自宅からMIFESやVZを持ってこようと思ったが，HDの中身をエコロジーで覗いていたら、余計なことをと言われ、頭にきたのでEDLINを使いこなし、教祖様と呼ばれた苦い思い出を思いだした。
COBOL系の技術者はパソコンにうとくて困る。
PS．MS-DOS版viを手にいれて、もっぱらこればかり使っています。（1ヶ月前から）
ATOKと使えばPFキーをまったく使わないで、打てるので、ブラインドタッチをする人には最適！！

トラトラトラ

■恐るべき新人！

うちの新人さんのお話しです。／Hirolin
平成03年08月20日（火）　午後04時41分16秒

私：「このフロッピーにあっちのパソコンからファイルをコピーしてきて！」
新人：「どのファイルですか？」
　（ここまでは合格かな）
私：「え――とね・・　Aドライブの SYSTEMの下のもの全部！」
新人：「どうやるんですか？」
私：「こないだやってみせただろ！」
新人：「わすれました」
私：「しょうがないな　COPY A:¥SYSTEM C:　とうつんだよ」
新人：「はいわかりました」
　そうして何とかファイルをコピーしてきました。
　そして次の日・・・
私：「このフロッピーにあっちのパソコンからBドライブのVZのしたのCENV.DEFというファイルをコピーしてきて！」
新人：「え――と　どうやるんでしたっけ？？」
　（ムカッ！）
私：「昨日おしえたばかりだろ！！」
新人：「あっ！　昨日とおなじでいいんですか」
私：「そうだよ！　やり方は一緒，ファイル名だけをかえればいいんだよ」
新人：「はいわかりました」
　しばらくして・・
　フロッピーをのぞいてみると・・なんと昨日と同じファイルばかり・・
私：「あれー！　昨日のファイルしかないぞ――　おまえ何をやったんだ？」
新人：「えっ！　昨日と同じですけど」
　なんとコマンドまで同じものをうったようです・・
　そこで、ファイルとは・・ファイル名とは・・COPYコマンドのパラメーター・・等などまたまたおしえてあげると・・
新人：「ファイルが見つかりませんってでてくるんですけど・・」
私：「そんなわけないだろ！　さっきまで使ってたんだ！！」
新人：「いや・・でもでてきたんですよ・・」
　しかたなくそのパソコンのところにいってみると画面には

```
A:¥USR> COPY B:CENV.DEF C:
B:TCENV.DEF ファイルが見つりません
            0 個のファイルをコピーしました
```

とでているではありませんか・・・（絶句）
う――ん、やっぱりいちいち打つコマンドを教えなきゃいけないのかな・・？
　新人の扱いに今日も頭を悩ませるHIROLINでした

■SE-NETシステムバージョアップのお知らせ

10月25日（金）よりSE-NETのホストマシンおよびホストプログラムをバージョンアップしました．これにより，旧来のSE-NETのIDはすべて抹消されました．SE-NETをご利用になる際には，新たにオンラインサインアップを行ってください．また，回線混雑緩和のためにGUESTによるダウンロードが禁止となりました．

以下に旧SE-NETと新しいSE-NETの違いを示しておきますので，アクセス時の参考にしてください．

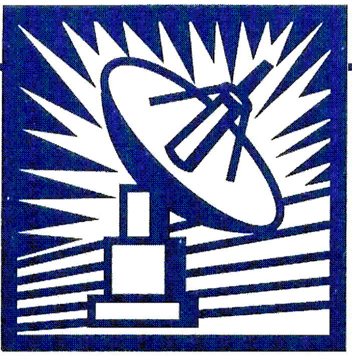
1．ハンドル名が全角に対応しました．
2．ZMODEMプロトコルを使用したアップロード／ダウンロードが可能に（一般的なファイル転送プロトコルすべてに対応）なりました．
3．電話代を節約するためのアクセスシミュレータを準備（「HGx」と呼ばれるアクセスシミュレータを準備することでメールの処理／メッセージの読み書き／ダウンロードファイルの指定がすべてオフラインで可能になりました）しました．
4．回線混雑緩和のため，時刻によって接続可能時間が変更になります．

また，今回のシステムアップに伴い，川畑一（Jairo）・山本和信（N.Yuki）の両氏に多大のご協力を願いました．編集部一同より，紙面を借りてお礼申し上げます．どうも，ありがとうございました．

●オンラインサインアップの前に

・お名前やご住所を漢字で登録する必要がありますので，かならず，日本語FEPが使える状態でアクセスしてください．
・管理者に対して利用目的などを申請する電子メールを出す必要があります．またファイルのダウンロードにはプロフィール（自己紹介）の登録が必須です．あらかじめエディタで作成し，準備してください．
・登録直後は仮登録となり，接続可能時間や一部のボードの読み書きなどに制限がありますが，2・3日で正規登録しますでお待ちください．
・IDの発行数に限りがありますので1カ月以上アクセスがないときは，登録を削除させていただきますので，ご了承ください．

Illustration : NOBUKI FUKUHARA



西川和久の
希望的観測

VGA vs 独自640×480 NEC，東芝，IBMの新製品を考える

1 640×480の画面を用意したマルチメディア対応PC-98GSだが

噂の画面解像度640×480のマシンがついにNECから発表された．「マルチメディア対応」「CD-ROM標準搭載」など，「これでもか!!」といろいろなキャッチフレーズがカタログに満載されている．これで値段が従来のPC-98とそれほど変わらなければ一気に……と思った私が馬鹿だった．386SX/20MHzを搭載したいちばん安いモデルでさえ，698,000円もするのだ．メモリは，たった2.6MB．日本語WindowsとMMEを動かすには少なすぎる．メモリを増設し，HDDを拡張してサウンドカードなどMMEに必要なボード類を増設すると軽く百万円を突破してしまう．いったい誰のために開発されたマシンなのだろうか!?　値段から考えても，単純に日本語Windowsを使いたいエンドユーザー狙いとは考えられないし，企業ユースとしても難しい．ターゲットはどこにあるのか？　私なりにいろいろ考えたみたが，答が得られない．

フッと思いついたのだが，NECお得意のアレである．「マルチメディアもサポートしています」と，何でもサポートしている安心感をユーザーに与える存在なのだ．「参加することに意義があるマシン」……それがPC-98GSである．このことを記者たちは察知したのか，業界紙を含む翌朝の新聞の新製品覧には，TFTカラーのラップトップPC-9801NCしか載っていなかった．その他の新製品にはまったく触れていないのだ．同時に発表があったものとしては，「N5200ラップトップ」「CRT一体型PC-9801CS」「MS-DOS Ver.5.0」「ポケットモデム」など，それだけでもニュースになるものがあったにも関わらず……．

もうひとつの考え方としては，型番がPC-9801GSではなく，PC-98GSということだ．これまでPC-98xxとしていろいろなハードウェアが発表されたが，どれも消えていく運命を背負ったマシンばかりであった．本家本元，本流ではない型番，それがPC-98xxシリーズである．今回のPC-98GSも残念ながらこのパターンなので，いずれ忘れ去られてしまうのか……．

2 DynaBook Vシリーズ 3モードのDOS5サポートで登場

こちらも「いつ出るか!?」とカウントダウン状態にあったマシン……VGA搭載DynaBook Vシリーズである．誰もが容易に予想できたことであろう．しかし私の予想を遥かに上回る完成度で登場したのだ．A4ジャストサイズ，HDD 80MB，日本語MS-DOS5.0など，ここまでは予想通りであった．しかし驚いたことに，この日本語MS-DOS 5.0では「J-3100」「PC-DOS」「DOS/V」3つのアプリケーションがすべて動くマルチ環境になっているのだ．VGAが搭載されたことで「ユーザーが勝手にDOS/Vをインストールして使うのは自由」というスタンスを取るものとばかり思っていたが，こんな早

ワンラインNEWS
富士通はパソコン用RDBを手掛ける3社と提携した．同社が提案する「MASSEAGE 90s」のソフト製品群に取り入れ，汎用機とパソコンが連動できる業務範囲を広げる．

い次期に３つの環境を統合化して正式サポートに踏み切ったのには恐れ入った．これでDOS/V陣営がかなり活発化するのは，誰の目から見ても明らかである．

連載第１回目で「AXとJ-3100も手を結んでくれれば……」と書いたが，AX側のそれに比べて予想以上に東芝側の対応が早かった．現在，PC互換機メーカーの動きも活発であるし，COMPAQまでもが上陸した．これは本当に面白くなってきた．

しかし，発表資料の中で気になる記述がある「IBM DOS/V」と「OADG DOS/V」の区別がなされているところだ．DynaBook VではOADG DOS/Vと互換性はあるが，IBM DOS/Vとは異なるような表記がされている．これは困った!! DOS/Vといっても２種類あるのか，また市場が混乱する……発表資料を読んだ人ならそう思ったハズだ．これについてはご心配なく．ある有力な関係筋からの情報によると，OADG DOS/VとはDOS5がベースになっているが，現状のIBM DOS/VはDOS4であり「IBM DOS J5.0/Vが正式発表される前にIBM DOS J5.0/Vに対応している」とはいえなかった……ただそれだけの理由だそうだ．面白いことにIBM J5.0/Vの発表は同日の数時間後にあったのだ．発表会場までもがすぐ近く……水面下ではいろいろな動きがあるようだ．:-)

3 XGA搭載PS/55Z 30Uは「ビッグMac」と呼ばれる斬新さ

DynaBook V発表の数時間後，日本IBMの発表があった．まず目を引くのがPS/55Z 30Uである．今までのOneBoxタイプと見た目のデザインがかなり変更され，パソコン通信などでは「ビッグMAC」と呼ばれだしている．好き嫌いがはっきり分かれそうな，IBMとしては斬新なデザインなのだ．386SX/20MHz，2MB RAM，13インチ高解像度対応フラットスクエアCRT，XGAを標準搭載しFDDモデルが425,000円，これは安い．XGA単体のカードが135,000円であることを考えるとかなりがんばった価格である．発表会場でいろいろなベンチマークテストプログラムを動かしてみたが，一般的な386SX/20MHzマシンと比較してもかなり高速である．

新しいPS/55 noteは，ユーザーの意見を取り入れかなり改善された．CPUのパワーアップ，ハードウェア液晶反転SW，外部CRT，３モードFDD，HDDモデルでもFDDは内蔵，外部キーボード接続可能など，バッテリー駆動時間を除けば，従来モデルの欠点はすべて解決している．また，外部CRTとキーボードが接続できるため，ラップトップではなく箱型コンピュータ（省スペース）としても使う用途も考え縦型スタンドまでもが用意されていた．ここまですると悪のりかな!? :-)

同時にIBM DOS J5.0/Vも発表があった．出荷開始は10月末である．発表自体はNECのほうが先だったが，実際の出荷は日本IBMのほうがかなり早い．386SX/16MHzのATバスマシン5510Sも248,000円という低価格で登場し，本気を感じさせる日本IBMの発表であった．

4 データショーで勢ぞろいするVGA搭載マシン．そして……

10月22日から開催されるデータショーは，ここで紹介したほかにもいろいろなニューモデルがところ狭しと展示されるであろう．そして日本IBM，東芝，AX陣営を含め，そのほとんどがVGAマシンだと予想できる．大手の非VGAマシンメーカーとしてはNEC，富士通，EPSON程度ではないだろうか!?こんな状況を去年の同次期に予想できた人は何人いただろうか？　来年はVGAをキーワードに，「非IBM-PCアーキテクチャ・マシン崩壊劇」がはじまるか……？

コンピュータの法律

他人にプログラムを作成させた場合の著作権の帰属　その1

◆

熊谷信太郎(弁護士)

■3通りの著作権の帰属

他人にプログラムを作成させた場合に，プログラム著作権が誰に帰属するかについては，

(1)他人に指図してプログラムを作成させた者にプログラム著作権が帰属する場合
(2)実際にプログラムの作成に携わった者にプログラム著作権が帰属する場合
(3)両者に著作権が帰属する場合

の3通りのケースがあります．

■プログラムは公表の有無を問わず法人に帰属

まず，(1)の他人を指図してプログラムを作成させた者にプログラム著作権が帰属する場合の典型例としては，会社などの法人[*1]が社員などの業務従事者に命じてプログラムを作成させる場合をあげることができます．

まず，プログラムに限らず，著作物全般について，著作権法は，法人の従業者が職務上作成した著作物で，法人名義で公表する場合には，個々の作成者ではなく，法人に著作権が帰属すると定めています[*2]．

これに対し，プログラムの場合には，一般の著作物の場合よりも，さらに法人に著作権が帰属する場合を広げて，法人の従業者が職務上作成したプログラムは，法人名義で公表しない場合でも，個々の作成者ではなく，法人に著作権が帰属すると定めています[*3]．

この両者の違いは，一般の著作物では法人名義で公表するものに限って法人に著作権が帰属するとしているのに対し，プログラムの場合には，外部に公表されないで使用されることが多いために，そのような制限を設けず，公表の有無を問わずに，法人の従業員が職務上作成したプログラムは，会社などの法人に著作権が帰属するとしている点にあります．

このように，会社の従業員が，職務上プログラムを作成した場合，その人にはプログラム著作権が発生せず，会社にプログラム著作権が帰属します．

したがって，会社に在職中に，職務上作成したプログラムを勝手に複製したり，また，退職後にそのプログラムを自己の作成したものとして売るなどの行為は，会社の著作権を侵害することになり，許されません．

もっとも，これらは法の定める原則であって，会社とプログラム作成者の契約によって，あるいは，会社の就業規則によって，これと異なる扱いをすることは許されていますので，自分の作成したプログラムについて著作権を設けて欲しい場合には，その旨を会社と契約しておく必要があります．

また，法人に著作権が帰属するための要件として，「法人の業務に従事する者」が作成したということがあげられていますが，これはいわゆる正社員と呼ばれる従業員のほか，会社の役員も含まれますし，そのほかパートタイマー，アルバイト，派遣労働者など，名称の如何を問わず，会社の指示に従い業務を遂行する者はすべて含まれます．

これに対し，「従事する者」は，人だけを指すと考えられるので，法人は含まれません．従って，プログラムの作成を下請会社に発注したような場合は，この規定には当たらず，プログラムの著作権がどちらに帰属するかは，契約内容により決定されることになります．

■指図した者に著作権が帰属する場合

このように，法人が社員に命じてプログラムを作成させる場合が，(1)の「他人を指図してプログラムを作成させた者にプログラム著作権が帰属する場合」の典型例といえるでしょうが，これ以外にも，(1)にあたる場合がありえます．

たとえば，人が人にプログラム作成を命ずる場合でも，プログラム作成を指図した者が，実際に創作に携わった者に対し，具体的な指示を与え，その意のままにあたかもその手足のごとく

ワンラインNEWS
バックスはPC-9801対応の日本語入力FEP「VJE-β」をVer.3.0にバージョンアップした．MS-DOS 5.0のタスクスイッチ機能に対応している．

他人を使って著作物を完成させたような場合には，指図した者だけにプログラム著作権が帰属すると考えられます．この場合には，実際に創作に携わった者が自らの創意工夫によってプログラムを創作したとはまったく評価できないので，その者にはプログラム著作権が発生しないのです．

他人にプログラムを作成させた場合の問題点の残りは，次回に検討します．

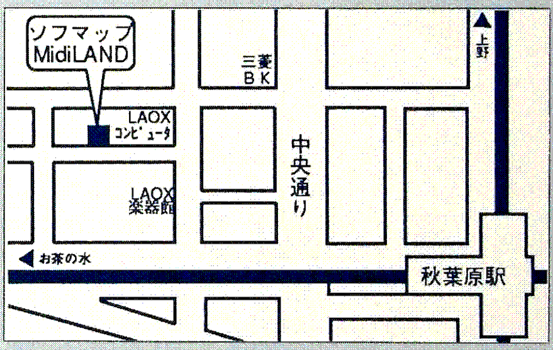
＊1　法律上，人のことを自然人と呼び，例外なく権利義務の帰属主体とされていますが，各種の団体については，すべてではなく法律が認めたものだけが権利義務の帰属主体とされ，それを法人と呼びます．
＊2　著作権法第15条第1項
「法人その他使用者の発意に基づきその法人等の業務に従事する者が職務上作成する著作物（プログラムの著作物を除く）で，その法人等が自己の著作の名義の下に公表するものの著作者は，その作成時における契約，勤務規則その他に別段の定めがない限り，その法人等とする．」
＊3　著作権法第15条第2項
「法人等の発意に基づきその法人等の業務に従事する者が職務上作成するプログラムの著作物の著作者は，その作成時における契約，勤務規則その他に別段の定めがない限り，その法人等とする．」

アキバ短信

ソフマップMidiLAND

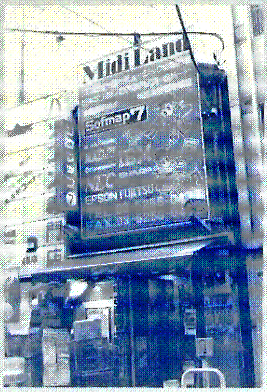
コンピュータミュージック専門店

営業時間　平日　11:50〜8:00　土曜　11:00〜8:00
日曜／祭日　10:00〜8:00　年中無休
TEL　03-5256-0477



秋葉原はいまやちょっとした建築・改装ブームだ．先月号で紹介した万世の新ビルもそのひとつだが，アキハバラデパートでも，先頃大改装が行われた．駅から離れたところでは，安売り王マヤ電気が古い店舗を取り壊しての改装中．Macユーザーにおなじみのイケショップも改装して，一部の売り場を蔵前橋通りぞいに移転した（もっとも元の店も健在だから，これは拡張というべきか）．その一方，JR秋葉原駅前には広大な空き地が広がっていて，中央通りの華やかさとは裏腹に，辺りは昼もひっそりとしている．建築・改装ブームもこの青果市場跡地の開発を見通した一種の先行投資という見方もできないわけではない．大型店はより一層の売り場を拡張して集客能力を高め，専門店はより分化してそのカラーを鮮明にする．

ソフマップは，専門店の代表選手のひとつである．関東地区と大阪地区では，パソコンやファミコンを売りに来た人が行列を作っている場面を映したテレビCFがオンエア中であることも手伝ってか，「ソフマップ」といえば，中古パソコン，ソフトの買い取り・販売店のイメージが強いが，もちろん新品も取り扱っている．現在秋葉原には中古ハード・ソフトを扱う本店を筆頭に11の店舗と，サービスセンターがあって，店舗ごとに「新品パソコン」「中古ファミコン専門」「マック専門」などと色分けされているのが特徴だ．



中でもソフマップの7号店にあたる「MidiLAND」はユニークなコンセプトを持つ．名前からもわかるようにここはコンピュータミュージックの専門店である．そもそもコンピュータミュージックには，パソコンのほかにMIDIインターフェース，MIDI楽器，音楽ソフトという3種の神器が必要である．中にはローランドの「ミュージ郎」のようにはじめからその3つがセットになっていて，ヘッドホンひとつですぐに音楽が楽しめるものもあるが，自分で演奏もしようという場合には，電子ピアノや音源ユニットも必要だ．

MidiLANDの1階はコンピュータミュージックに必要な各種ハードウェア，ソフトウェアの売り場．2階はキーボードを中心とした中古電子楽器売り場．ここに来れば必要なものはひと通りそろう．MIDIインターフェースを標準装備したAtariSTもあれば，音源ユニットやMIDIインターフェース，PC-9801用，X68000用，Mac用，アミーガ用などの各種音楽ソフト，MIDIケーブルなども揃っている．すぐ近くにはLAOXの楽器館もあるから，コンピュータミュージックに興味・関心のある向きにはおすすめのエリアである．

ワンラインNEWS
シャープはノートブック型UNIXワークステーション「UN-10」を発売した．UNIXワークステーションとしては最小・最軽量を実現している．

●プログラマの仕事場 9

Programmer's Working Space

株式会社スタジオ・ゲン

CADソフトからマルチメディア新製品の企画・開発までこなす不思議なプロフェッショナル集団

スタジオ・ゲンは，ハードウェアとソフトウェアの企画開発を行うシステムハウスであるが，各種エレクトロニクス製品のデザイン，マーケティング，新製品開発も手がければ，ニューロコンピュータなどの新技術の調査・研究も行う．一見何のつながりもないように見えるこれらの事業を，少数精鋭のプロフェッショナルな技術者が展開している型破りな会社だ．同社をたずねて代表取締役の宮沢丈夫氏にお話をうかがった．

ソフトウェア開発の売上は全体の2分の1

スタジオ・ゲンのオフィスは，港区高輪の閑静な住宅街のとあるマンションの1室にある．といっても，狭い和室にパソコンがぎっしりと，煙草の煙がもうもうとたち上るといったところを想像していただいては困る．柔らかく抑えられた照明のせいか，室内の雰囲気はどことなく明るく穏やかだ．ちょうど私たちが訪れたときに，奥の部屋では外国からの来客をまじえて打ち合わせがすすめられていたが，ときおり聞こえてくる会話はすべて英語である．室内にはMacintoshあり，ワープロあり，パソコンあり，ビデオあり，大きなファンがうなりをあげるワークステーションもあるのだが，ソフトウェアの開発現場というよりは，何かぜいたくなおもちゃ箱の中に迷い込んだような錯覚をおこしてしまいそうだ．

スタジオ・ゲンの代表取締役である宮沢氏は，ASCII誌上で「現代科学の最先端をあなたのパソコンで」というタイトルで，遺伝的アルゴリズムやフラクタルなどの先端技術をパソコンのプログラムを通して体験するという，ユニークな試みを行った著者でもあるといえば，心あたりのある人もいるだろう．

宮沢氏は大学生のころまでは，ミュージシャン志望であったという．バンドを組んで，ヤマハのコンテストに出場，全国で2位になったほどの実力ではあったが，プロデビューには至らなかった．そこで，それまで音楽に向いていた関心が一挙にパソコンに向けられて，気がついたときはASCIIで原稿を書くまでになっていたのだそうだ．1983年にスタジオ・ゲンを設立．最初は宮沢氏と，デザイナー，営業担当の3名でスタートしたが，現在ではデザイン部門は独立して別会社になっている．

宮沢氏は同社の業務の内容を次のように語ってくれた．

「3分の1がCAD (Computer-Aided Design)．CADのソフトをワークステーションで作るということをやっています．マルチメディアが3分の1．残りの3分の1がその他ですが，その他のうち半分がエレクトロニクス製品に関するものです．これはメーカーさんの新製品の企画・開発をサポートする仕事で，具体的にいうと，たと

ワンラインNEWS
アライドテレシスはノート型パソコン用で，拡張ユニットが不要の専用拡張バスを利用するLANアダプタを発売した．エプソン製のノート型パソコンに対応している．

えばあるメーカーさんが『3万円以内でビジュアルな要素が入ったオモチャを作りたい』といった条件を出してきます．それに見合う製品のデザイン・機能などを提案するわけです．

それから，残りが新技術調査ですね．私自身ASCIIにニューラルネットの連載を3年ほど前にやりました．これは雑誌の記事としてはかなりレベルの高いもので，解説をした上で実際にC言語のプログラムを載せています．現在も現代科学の最先端をパソコンのプログラムを使って紹介する連載をしています．これがきっかけになって，新しい技術がメーカーさんでどのように応用できるかといった仕事ーファジー&ニューロというような製品にもいくつか関わっています．ソフトウェアハウスとしての売上は全体の半分くらいなんですよ．」

高いクオリティを求めてシンガポールで求人

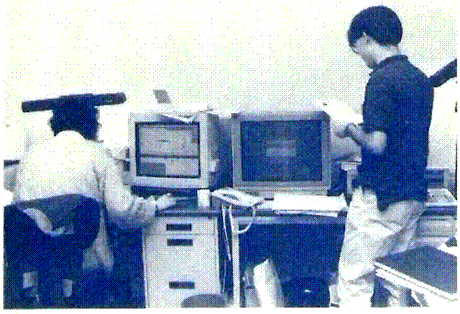
スタジオ・ゲンのスタッフは現在12名だが，その顔ぶれは国際色豊かだ．「アメリカ人，シンガポール人，マレーシア人，香港人，つい最近まではイスラエル人もいたんですよ．皆プログラマやSEです」（宮沢氏）．外国人が多いのは，仕事の内容との関連なのだろうかと尋ねてみると，意外な答がかえってきた．

「スタッフの中には大学でコンピュータ関係の博士課程を出てきた者もいるのですが，必要なだけの高いクォリティの人間が手近にいなかったこと．それからASCIIの取材の仕事でシンガポールに行ったときに，シンガポールのいろいろな会社を回ったのですが，そのときすごくレベルが高いと感じたのです．僕をはじめ会社のスタッフは，たまたま皆英語には不自由していなかったので，何もそれなら日本人にこだわることはないんじゃないかと，シンガポールで求人広告を出して，シンガポールまで行って面接した上で選んできました．ですから，外国人を採用したのは安い労働力を得たいからではなくて，きわめて優秀な人を採用したかったからだというのがいちばんの理由ですね．」

外国まで出かけていって人を探してくるというだけあって，仕事上は高い技術と知識が求められる．

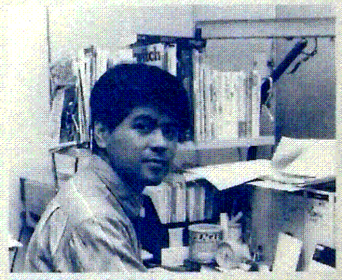
「たとえば，今度MITのAIラボの先生とジョイントして進めようとしているプロジェクトがあるのですが，それも僕とシンガポール人と2人でアメリカへ行って話をつけてきました．」

これなら社内で英語の打ち合わせが行われているのも当然だろう．



About Company

株式会社スタジオ・ゲン．1983年5月設立．代表取締役は宮沢丈夫氏．業務内容はUNIX上でのCAD開発からDVIによるマルチメディアのタイトル作成，エレクトロニクス製品に関する新製品の開発企画，新技術調査など多彩な分野にわたる．スタッフ12人中8人が開発に携わる．



代表取締役　宮沢丈夫氏

Location

オフィスは港区高輪，品川駅側から見て新高輪プリンスホテルの裏手にあたる一角にある．辺りは閑静な高級住宅街．

Products

マルチポートルータ「SmileTalk-IRI」（本体標準価格298,000円）

イーサネットとローカルトークを接続して，UNIXワークステーション環境とMacintosh環境を結合し，相互にデータのやり取り（電子メール）を行ったり，Macintosh側のレーザーライターやカラースキャナ，ネットモデム等をワークステーション側から使用可能にする．

また，高速シリアルポートを標準装備しているため，別売のターミナルアダプタ（ST-64TA）を接続して，INSネット64の高速デジタル伝送機能を利用した広域ネットワーク（WAN）の構築も可能である．

DVI

DVI（Digital Video Interactive）は，パソコン上で動画，静止画，音声を扱うマルチメディア技術．デジタル画像データはアナログデータと違って，動画や静止画をさまざまにパソコン上で加工できるという特徴がある．デジタル画像データは，記憶

媒体を大量に消費するが，DVIではVDP(Video Display Processor)と呼ばれる高機能で小型のチップを使って，画像データを圧縮し，CD-ROM 1枚に最大約72分のフルスクリーン動画をおさめることができる．

DVIの開発は，1983年にRCAの研究所ではじまったが，現在この技術はインテル社に移行しており，同社ではビデオプロセッサチップ，ボード製品，アプリケーション開発システムやツールなど各種ハードウェア・ソフトウェア製品を提供している．

応用例としては，電子カタログ的なもの，企業や学校などでの教育システムなどがある．画像処理はプログラマブルであるため，今後の技術的可能性は大きく，さらに応用範囲は広がると思われる．10月に行われた「マルチメディア'91」では，記憶媒体にCD-ROMを使ったDVIソフトを再生するDVIプレーヤーも数社から出品されている．

マルチメディア市場が立ち上がる3年後を見つめて

マルチメディアとは，映像，画像，音声，文字，データなどのさまざまな素材を複合化して利用することをいうのだが，最近はあらゆるところで『マルチメディア』という言葉だけがひとり歩きをしていて，かなり漠然とした印象がある．宮沢氏は同社で作成したマルチメディアのデモを紹介してくれた．

最初の画面には世界地図が表示されている．マウスをクリックするたびにその地域の地図がどんどん拡大表示されていく．画面上に日本地図が表示されたところで「東京」をクリックすると，今度は東京の地図が，その中からさらに「渋谷」を選ぶと渋谷の地図が出てくる．ボタンを押すと今度は街頭の写真が出てくる．「この道を左に曲がると……」という宮沢氏の説明で，切り換わった画面をよく見てみると，今度は左に曲がった先の街角の写真が表示されている．つまり，画面の中で実際に渋谷の街を歩いてみることができるのである．このシステムの開発には，DVIを使ったインテル社の開発用システムが使われているそうだ．

宮沢氏はスタジオ・ゲンのマルチメディア事業についてこう語っている．

「マルチメディアをやっているといっても，日本にはまだマルチメディアのマーケットが全然ありませんから，いまはまだメーカーさんがこういうハードを作るからというお話があると，共同でハードウェアやその上で動くソフトウェアを作っていくといったところがメインです．あと3年くらいして，CD-ROMがパソコンにも標準で付くようになってくればうちの出番があるかなと思いますが，それまではどうしてもお客様の仕事をしたり，スペシャルなハードウェア用にいろいろなデモを作るという仕事がほとんどですね．いまは力を蓄えているところです．」

マルチメディアの「ビジカルク」「一太郎」を

「新技術の調査・研究の仕事にも力を入れているのは，いまがうちとしてはまだ勉強中の期間だからということがあります．うちがめざしている会社になって，いいものを作り出すにはあと2～3年かかるでしょう．そのときには，マルチメディアや新しい技術，ニューロコンピュータなど，いま手がけているいろいろなものを全部統合したものを作りたいのです．あと2～3年してマルチメディアのマーケットがたちあがるときまでに，うちでいまやっているコンピュータグラフィックスとか，マルチメディアとか，ニューテクノロジーとか，最先端の研究とかを統合していく必要があるでしょう．マルチメディアが，ただ単に絵が出て，音が出て，ビデオが出ますというだけではおもしろくないので，もっと2歩も3歩もすすんだものを作りたいなと考えています．」

「たとえば『ターミネーター2』をご覧になった方はわかると思いますが，あそこで使われているCGというのは，理論的に最先端の研究成果を非常にローエンドな消費者レベルに落として見せているわけです．マルチメディアはずいぶんと騒がれていますが，起爆剤となるソフトがないのです．マルチメディアにおける（ビジネスソフトのさきがけとしての）ビジカルクみたいなものがあるとしたら，またマルチメディアに（MS-DOSパソコン普及のきっかけとなった）一太郎に相当するものがあるとしたら，そういったアプリケーションを作りたいのです．そのためには，コンピュータグラフィックスをはじめいろいろな技術を統合する必要があるのです．」

マルチメディア市場を確立する起爆剤としてのインパクトをもったソフトウェアを作り出すことが宮沢氏の目標であるようだ．それは，「テレビゲームよりも，ビデオ映画よりも面白いものでなければならない．先日開かれたマルチメディア'91を見て，ゲームや映画より感激した人はひとりもいないはずです．マルチメディアは，まだまだこれからのものです．数年後に市場として立ち上がってくれれば，と考えています」と，宮沢氏．

多様な才能を集めたスタジオ・ゲン，マルチメディアというプラットホームでいったいどういう世界を提示してくれるのか，注目していきたい．

ワンラインNEWS
米クアンタムコーポレーションは記憶容量が130Mバイトと最大の容量をもつ2.5インチHDDを開発した．これまでは富士通製の90Mバイトが2.5インチHDDでは最大だった．

information・page

●イベント

■11月28日(木)～11月30日(土)
NECパソコンフェア'91
日本電気と周辺機器・ソフトウェアメーカーによる展示会。12月13日(金)・14日(土)はインテックス大阪(大阪市)、12月20日(金)・21日(土)は中小企業振興会館(名古屋市)でも開催される。
会場：日本コンベンションセンター(幕張メッセ) 入場料：無料
問い合わせ先：日本電気(株) TEL 03-3454-1111

■12月3日(火)～12月4日(水)
UNIX Fair'91
UNIX関連製品の展示とセミナーが行われる。
会場：新宿NSビル地下イベントホールほか 入場料：無料
問い合わせ先：IDGワールドエキスポ・ジャパン TEL 03-5276-3751

●海外イベント

■12月9日(月)～12月11日(水)
PC WINDOWS/San Francisco
Windows関連製品の展示とコンファレンスが行われる。
12月9日は招待者のみ
会場：Moscone Center(米国・カリフォルニア州サンフランシスコ)
問い合わせ先：IDGワールドエキスポ・ジャパン TEL 03-5276-3751

●セミナー

■11月21日(木)
活用されるためのCASE 構造化手法実践セミナーin大阪
構造化分析・設計の全体図、リアルタイム構造化手法などをKangaToolを用いて解説する。
会場：大阪駅前第4ビル20F(大阪市・北区)
受講料：30,900円(消費税・資料・昼食代含)
問い合わせ先：(株)ライフボート「活用されるためのCASE─構造化手法実践セミナーin大阪」係 TEL 03-3293-4714

■11月22日(金)
リアルタイム制御プログラミングセミナーin大阪
リアルタイムシステムの構築方法、MS-DOSシステムからリアルタイムシステムへの拡張方法などをLattice C/RTシリーズを例に解説する。
会場：大阪駅前第4ビル20F(大阪市・北区)
受講料：30,900円(消費税・資料・昼食代含)
問い合わせ先：(株)ライフボート「リアルタイム制御プログラミングセミナーin大阪」係 TEL 03-3293-3887

■11月26日(火)～11月28日(木)
オブジェクト指向設計とC++セミナー
オブジェクト指向技術によるC++プログラミングを体験しながら学習する。
会場：虎の門パストラル(東京)
受講料：190,000円(3名以上、2講座受講者に割引あり)
問い合わせ先：ソフトウェアテクノロジーインターナショナル TEL 03-5689-3536

■12月5日(木)～12月6日(金)
オブジェクトオリエンテッドテクノロジー入門セミナー
オブジェクト指向による分析設計プログラミングからその長所短所までを解説する。
会場：麹町会館(東京)
受講料：96,000円(3名以上割引あり)
問い合わせ先：ソフトウェアテクノロジーインターナショナル TEL 03-5689-3536

Microsoft University

■12月2日(月)～12月3日(火)
C言語入門
QuickCでC言語の入門コース。
受講料：60,000円(2日間)

■11月25日(月)～11月29日(金)、12月16日(月)～12月20日(金)
Microsoft Windows Applications Programming
Windowsアプリケーション開発のための上級コース。
受講料：180,000円(5日間)

■11月25日(月)～11月29日(金)、12月16日(月)～12月20日(金)
MS OS/2 Presentation Manager Applications Programming
OS/2 PMアプリケーション開発のための上級コース。
受講料：180,000円(5日間)

■11月25日(月)～11月29日(金)、12月16日(月)～12月20日(金)
Microsoft LAN Manager Applications Programming
LANマネージャアプリケーション開発のための上級コース。
受講料：180,000円(5日間)

■12月2日(月)～12月6日(金)
MS OS/2 Presentation Manager Programming Environment
PMアプリケーション開発のための入門コース。
受講料：180,000円(5日間)

■12月2日(月)～12月3日(火)
QuickBASIC初級
QuickBASICを使ったBASICプログラミングの入門コース。
受講料：60,000円(2日間)

■12月4日(水)～12月6日(金)
QuickBASIC中級
QuickBASICを活用するための本格的なプログラミングを中心としたコース。
受講料：90,000円(3日間)

■12月9日(月)～12月11日(水)
i8086基本コース
8086、8088のアーキテクチャなどを実習を交えて解説する。
受講料：90,000円(3日間)

■12月9日(月)～12月13日(金)
i8086システム
8086上でのプログラミング開発技法を修得するためのコース。
受講料：150,000円(5日間)

■12月11日(水)～12月13日(金)
X-Windowプログラミング
X-Window上でプログラム開発を行うための基本コース。
受講料：90,000円(3日間)

■12月9日(月)～12月13日(金)
Microsoft Windows Programming Environment
Windowsアプリケーション開発のための入門コース。
受講料：180,000円(5日間)

■12月10日(火)～12月13日(金)
Microsoft LAN Manager Programming Environment
LANマネージャアプリケーション開発のための入門コース。
受講料：150,000円(4日間)

■12月11日(水)～12月13日(金)
C言語応用
MS-C 6.0を使ってC言語の高度な技術を学ぶ実習中心のコース。
受講料：90,000円(3日間)

■12月16日(月)～12月17日(火)
i80386/80486アーキテクチャ
386システム設計者のための導入コース。
受講料：90,000円(2日間)

■12月16日(月)～12月20日(金)
i80386/80486システム
80386/486のアーキテクチャとプログラム開発手法を修得するためのコース。
受講料：220,000円(5日間)

■12月17日(火)～12月20日(金)
MS OS/2 Device Drivers
MS OS/2のデバイスドライバの機能と構造を解説しサンプルの作成を行う。
受講料：150,000円(4日間)

Microsoft Universityは、OS/2、Windowsアプリケーション開発者のために定期的に開催されているセミナーです。
会場は三田会場(東京都港区)または特別会場(別途通知)です。
問い合わせ先：富士ソフトウェア(株)MU教育センター TEL 03-3452-7

NEWS & NEW PRODUCTS

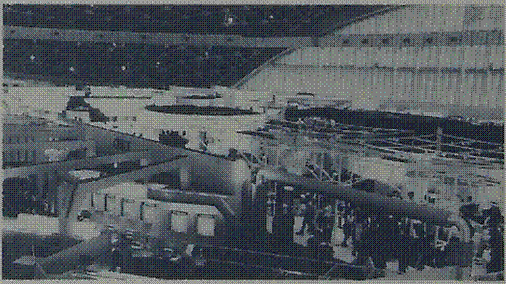
データショウ'91，160余社が参加して開催される

秋の恒例行事となっている「データショウ'91」が東京・晴海の国際見本市会場おいて，10月21日から25日までの４日間にわたって開催された．春のビジネスショウと並んで小型コンピュータ関連のハードウェア，ソフトウェアが一堂に会する場であるだけに，連日多くの人出でにぎわった．最近では，ショウの前に各メーカーから新製品が発表されるのが通例になり，会場で未発表の新製品に出会えることは少なくなっているが，最新のコンピューティング環境を肌で感じられることが多くの人を集めている要因だろう．

パソコンハードウェアは，昨年来，ノートブック型が主流となり，機種ラインアップのなかで重要な位置を占めるようになってきた．日本電気は，カラー液晶ディスプレイを装備したノート型の新機種PC-9801NCを体験コーナーを設けてアピール，シャープも参考出品ながらAT互換のカラーノートブック型パソコンを前面に出していた．また，ノートブック型の先駆けとなった東芝のダイナブックは，DOS/V対応のＶシリーズをまじえた展示で，注目を集めていた．また，DOS/Vの発信元である日本アイ・ビー・エムは，DOS/Vマシンを勢ぞろいさせるとともに，OS/2 V2をOS/2，DOS，Windowsなどのアプリケーションが動作するプラットホームとして紹介していた．ショウ直前にノートブック型マッキントッシュを発表したアップルも多くの人を集めた．

システム環境の統合化ということも今回の特色といえる．日本電気は，「プラットホーム98構想」のもとに，N5200・システム3100シリーズに98のソフトが動作する機種を投入した．東芝のダイナブックＶシリーズでは，J-3100・DOS/V・PC-DOS対応のソフトが動作する．前述のOS/2 V2のようなOSに，あるいはMS-DOSの機能などによって，動作するアプリケーションの数を増やすという流れが進みつつある．

一方，SISやLANなど，より重要な実務処理をパーソナルシステムに移行しようという動きが強まるなかで，活気づいているのはUNIXやマッキントッシュだ．もともとネットワーク環境を備えたこれらのコンピュータを，ビジネスマーケットに浸透させようという動きが盛んだ．サン・マイクロシステムズのようなワークステーションメーカーが，EWSからビジネスへと市場を拡大しようとしている．



日本IBM，次世代の統合プラットフォーム「OS/2 J2.0」を発売

日本アイ・ビー・エム（株）（TEL 03-3586-1111）は，エンドユーザー中心のコンピュータ時代の中核となる次世代OSとして「IBMオペレーティングシステム／２バージョンJ2.0（OS/2 J2.0）」を発売した．「OS/2 J2.0」は，DOS，DOS/V，Windows，OS/2（16ビット・32ビット）で稼働するすべてのアプリケーションを，ひとつの画面上で同時に稼働することができる統合プラットフォームで，MVDM（Multiple Virtual Dos Machine）機能により，数十個のDOSを同時に稼働させたものと同様な環境を実現することができる．オブジェクト指向技術に基づいた親しみやすいグラフィカルユーザーインターフェース「ワークプレース・シェル」を備え，現行の1024×768ドットの解像度に加え，VGA（640×480ドット）もサポートしている．価格は33,200円で，OS/2 J1.3ユーザーには無料でバージョンアップを行う．また同時に，80486（25MHz）を搭載し，OS/2に対応したハイエンドマシン「PS/55モデル5580-W1」（1,320,000円～1,420,000円），「同W0」（1,520,000円～2,070,000円），ネットワークOS「NetWare 386 V3.1J PS/55版」も発表した．

ワンラインNEWS
シスケンはプログラムのソースリストを編集し，プリントアウトする「PROLST（プロリスト）」を発売した．任意の用紙に好みの印字幅，行数でプリントアウトできる．

NEWS & NEW PRODUCTS

マイクロウェア・システムズ，バージョンコントロールシステム「PVCS」を発売

マイクロウェア・システムズ（株）（営業部：TEL 03-3257-9000）は，Sage Software社の開発したコンフィグレーションマネージメントのためのバージョンコントロールシステム「POLYTRON Version Control System（PVCS）」を発売した．PVCSは，あらゆる種類のファイル構造，作成過程，変更箇所などをきめ細かく管理でき，オブジェクトコード，実行可能コード，データベースなどのファイルと同様に，ソースコードファイルや標準テキストファイルなども管理できる．またパッケージには，プログラム作成作業を一括管理するPolyMakeも含まれる．OS-9バージョン2.4以降，OS-9000バージョン1.2以降の開発システムに対応．価格は150,000円．

伊澄システム製作所，多機種対応の低価格アセンブラを発売

（有）伊澄システム製作所（TEL 03-3488-5243）は，コンピュータの機種を問わず動作するMS-DOS用「Lightマクロアセンブラ」を発売した．製品には，マクロアセンブラ，リンカ，ソースレベルデバッガが含まれている．アセンブラはマルチパス方式の最適化アセンブラで，前方参照を正確に解決し，ジャンプ命令の最適化が行える．リンカは，EXE形式ファイルだけでなく，COM形式ファイルも直接作成することができる．デバッガは，ソースコードの表示やアドレスのラベル名による指定が可能で，簡単に実行地点の追跡表示のみを行うこともできる．価格は9,800円．

日本IBM，日本初のXGA搭載デスクトップパソコンほかを発売

日本アイ・ビー・エム（株）（TEL 03-3586-1111）は，XGAを搭載したパソコンを発売した．XGAは，1024×768で256色の表示が可能となる．XGAを搭載したのは「PS/55Zシリーズモデル5530-U」と「同5510-T」，ともにMCA（マイクロチャネルアーキテクチャ）を採用．「5530-U」（写真）は，13インチディスプレイとの一体型，「5510-T」は，同シリーズの最上位モデルでデスクトップタイプ．同時に，VGA搭載の「PS/55Zシリーズモデル5510-S」（248,000円～465,000円），「PS/55noteモデルN23sx」（352,000円～458,000円），「同N51sx」（398,000円～414,000円）も発表された．「5510-S」は，5510-Zの上位機にあたり，CPUに80386SX（16MHz）を搭載したATバスのモデル，「PS/55noteモデルN23sx」は，80MBハードディスクを搭載，「同N51sx」は，MCAを採用している．日本語DOSの強化版として「DOSバージョンJ5.0」，「DOSバージョンJ5.0/V（DOS/V）」も発表した．

■PS/55Z　モデル5530-U01/UV1/UW4/UW8

プロセッサ：80386SX（20MHz）／メモリ：2MB～6MB（最大14MBまで拡張可能）／外部記憶装置：3.5インチFDD×1（U01/UV1），3.5インチFDD×1・40MB～80MB HDD×1（UW4/UW8）／価格：425,000円～728,000円

■PS/55Z　モデル5510-T08/TW8

プロセッサ：80386（20MHz）／メモリ：2MB～6MB（最大14MBまで拡張可能）／外部記憶装置：3.5インチFDD×1（T08），3.5インチFDD×1・80MB HDD×1（TW8）／価格：628,000円～751,000円

NEWS & NEW PRODUCTS

東芝，DOS/Vに対応したDynaBook Vシリーズなどを発売

（株）東芝（パソコン営業推進部：TEL 03-3457-2930）は，VGA対応の640×480ドット高精細ハイコントラスト白黒液晶ディスプレイを採用し，CPUに80386SL（25MHz）などを搭載した国内最高速の32ビットブック型パソコン「DynaBook Vシリーズ」など4機種12モデルを発売した．「DynaBook Vシリーズ」は，同時に発表された「日英MS-DOS V5.0」（23,000円）を搭載することにより，J-3100，DOS/V，IBM PCに対応したソフトが動作する．VGA対応の「DynaBook V386/25」「同386/20」のほか，80386SXを搭載した「DynaBook386/20」（278,000円～378,000円），「同386」（238,000円～33,8000円）がある．VGAを搭載していない新機種も，VGAアダプタとCRTを用意することでVシリーズと同様の機能を実現できる．

■ DynaBook V386/25（J-3100SL 001VW/041VW/081VW）

プロセッサ：80386SL（25MHz）／メモリ：2MB（10MBまで拡張可能）／外部記憶装置：3.5インチFDD×1（001VW），3.5インチFDD×1・40MB HDD×1（041VW），3.5インチFDD×1・80MB HDD×1（081VW）／ディスプレイ：16諧調FLサイドライト付白黒液晶／価格：378,000円～558,000円

■ DynaBook V386/20（J-3100SX 001VW/041VW/081VW）

プロセッサ：80386SX（20MHz）／メモリ：2MB（（0MBまで拡張可能）／外部記憶装置：3.5インチFDD×1（001VW），3.5インチFDD×1・40MB HDD×1（041VW），3.5インチFDD×1・80MB HDD×1（081VW）／ディスプレイ：16諧調FLサイドライト付白黒液晶／価格：328,000円～50,8000円

日本電気，カラー液晶搭載した98NOTE，マルチメディアパソコンなどを発売

日本電気（株），日本電気ホームエレクトロニクス（株）（NECパソコンインフォメーションセンター：TEL 03-3452-8000）は，TFTカラー液晶ディスプレイを搭載し，4096色中16色表示が可能なノート型パソコン「PC-9801NC（98NOTEカラー）」と，テレビやビデオの映像を編集，出力するデスクトップビデオ（DTV）に対応したマルチメディアパソコン「PC-98GS」を発売した．マルチメディア関連製品として，「マルチメディアエンジンセット」などのボード類，マルチメディア対応アプリケーションの開発・実行を行う「オーサウェアスター」も発売する．同時に，省スペースタイプのカラーCRTディスプレイ一体型の「PC-9801CS」（398,000円～675,000円）も発売した．

■PC-9801NC/NC40

プロセッサ：80386SX（20MHz）／メモリ：2.6MB（RAMドライブを増設メモリとして使用した場合3.6MB）／外部記憶装置：3.5インチFDD×1（NC），3.5インチFDD×1・40MB HDD×1（NC40）／ディスプレイ：TFTカラー液晶／価格：598,000円（NC），738,000円（NC40）

■PC-98GS model1/model2

プロセッサ：80386SX（20MHz）／メモリ：2.6MB／外部記憶装置：3.5インチFDD×2・40MB HDD×1（model2はCD-ROMを内蔵）／価格：698,000円（model1），828,000円（model2）

アップルコンピュータ，ノートブック型パソコンなど新機種を発表

アップルコンピュータジャパン（株）（TEL 03-3592-0505）は，同社初のノートブック型パソコン「Macintosh PowerBook」など，3機種14モデルを発表した．「Macintosh PowerBook」は，トラックボール，パームレスト，フルサイズキーボードを装備しているほか，SCSIポート，サウンド出力ポートなど，従来のMacintoshのもつ必要な機能はすべて内蔵している．「同Classic II」は，68030の搭載によりClassicの約2倍の性能を実現し，浮動小数点演算コプロセッサのサポートなどの拡張

ワンラインNEWS
日本電気コンピュータシステムはWindows3.0対応の通信ソフト「WINTERM Ver.2.0」を発売．他のアプリケーションとリアルタイムでデータの授受を行うDDE機能をもつ．

NEWS & NEW PRODUCTS

性を備え，マイクロフォンとサウンド入力機能も追加されている．「同Quadra」は，68040を搭載したハイエンドユーザー向けの最上位機で，24ビットカラーグラフィックス機能を内蔵して，構成の多様化と拡張性を重視した「同900」は，Ethernetネットワーク・SCSI/NuBusサブシステム，キーロックセキュリティ機能などを装備している．また同時に，現行のMacintosh LC，IIfxなどの値下げを行うことを発表した．

■Macintosh PowerBook100/140/170

プロセッサ：68000(100)，68030(140/170，170は数値演算コプロセッサ68882装備)／メモリ：2MB(100)，4MB(140/170)／外部記憶装置：20MB HDD×1(100，外部FD付タイプ有)，3.5インチFDD×1・40MB HDD×1(140/170)／価格：358,000円〜748,000円

■Macintosh Classic II

プロセッサ：68030／メモリ：4MB／外部記憶装置：3.5インチFDD×1(4MB/FD)，3.5インチFDD×1・40MB HDD×1(4MB/HD40)，3.5インチFDD×1・80MB HDD×1(4MB/HD80)／価格：298,000円〜458,000円

■Macintosh Quadra 700/900

プロセッサ：68040／メモリ：4MB／外部記憶装置：3.5インチFDD×1(4MB/FD)，3.5インチFDD×1・80MB〜400MB HDD×1／価格：1,238,000円〜1,678,000円(400MB HDD内蔵タイプの価格は未定)

エプソン，カラー液晶ディスプレイ搭載のブック型パソコンなどを発売

セイコーエプソン(株)，エプソン販売(株)(エプソンPCインフォメーションセンター：TEL 03-3377-3531)は，MIMアクティブマトリクス方式カラー液晶ディスプレイを搭載したフルカラー32ビットブックパソコン「エプソンPC-386BOOK LC」，NTN単純マトリクス方式カラー液晶ディスプレイを搭載し，カラーブックパソコンとしてははじめて50万円を切る価格を実現した「同LX」，80386SXを搭載し，30×30cmのコンパクトサイズを実現した「エプソンPC-386P」，40MBハードディスクパックが内蔵可能な32ビットノートパソコン「エプソンPC-386NOTE AE」を発売した．また同時に，「日本語MS-DOS Ver.5.0」(12,800円)を発表した．

■PC-386BOOK LX/LC

プロセッサ：80386SX(16MHz)／メモリ：2.6MB(最大8.6MBまで拡張可能)／外部記憶装置：3.5インチFDD×2／価格：498,000円(LX-2)，658,000円(LC-2)

■PC-386P

プロセッサ：80386SX(16MHz)／メモリ：1.6MB(最大14.6MBまで拡張可能)／外部記憶装置：3.5インチFDD×2／価格：268,000円

■PC-386NOTE AE

プロセッサ：80386SX(16MHz)／メモリ：640KB(最大8.6MBまで拡張可能)／外部記憶装置：3.5インチFDD×1・ICカードドライブ×1／価格：238,000円

IBM・アップル，テクノロジー分野での提携に合意

IBMとアップルコンピュータは，パーソナルコンピュータ産業における新時代の幕開けを宣言する包括的な合意を結んだことを発表した．新たな開発やテクノロジー，ソフトウェアのライセンス供与を通じて，大規模ネットワークでのMacintoshの利用をより容易にし，IBMのUNIX対応ソフトであるAIX，Macintoshインターフェース，POWERアーキテクチャーを基盤にした新しいオープン・システム環境「PowerOpen」を開発していく．また，両社とモトローラは，パーソナルコンピュータやワークステーションのエントリーモデルへの搭載に適した新しいRISCマイクロプロセッサー・ファミリーを開発する予定．この合意に基づき，アップルとIBMはマルチメディアの合弁会社「カレイダ」とオブジェクト指向ソフトウェアの合弁会社「タリジェント」を設立する．

Programmer's Program

Newton-98と同様の操作性をもつバックアップツール

Newton-Backup

けがや病気のときに補償してくれるのが生命保険，自動車事故で補償してくれるのが自動車保険，同様にハードディスクやパソコン本体には損害保険がかけられる．しかし，ハードディスクのデータを補償してくれる保険はない．バックアップは，そんなハードディスクのデータを守る唯一の手段である．

転ばぬ先のバックアップ

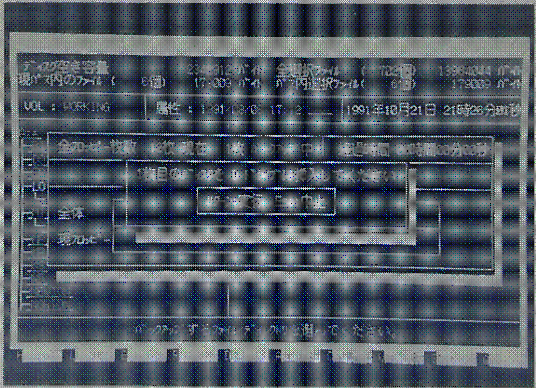
2年前，PC-9800シリーズも正式にSCSIタイプのハードディスクインターフェースがサポートされた．従来のインターフェースのように最大40MBのハードディスクを2台までという制限もなくなり，いよいよハードディスク大容量時代を迎えることとなった．いまなら，80MB程度の容量をもつハードディスクが10万円程度で購入できる．また，100MB以上に至っては，1MB当たりの単価も1,000円を切るハードディスクも存在する．

ハードディスクがあれば，大規模なアプリケーションを動作させても，フロッピーディスクの入れ換えをする必要もないし，数多くのアプリケーションをインストールしておくことができるので，作業効率も大幅に改善される．最近では，ハードディスクでの使用を前提としたアプリケーションも珍しくなくなってきている．とくに，開発環境にはなくてはならない周辺機器のひとつだろう．

しかし，このハードディスクもディスク自身は，フロッピーディスク同様に消耗品であることをわすれてはならない．しかも，フロッピーディスクに比べ数十倍の容量をもつため，失われるデータも膨大なものとなる．日頃からしっかりバックアップをしていないと，蓄積してきたライブラリやソースが，予告もなくある日消えてしまうことになりかねない．

RAM・ネットワークドライブも対応

MS-DOSのパッケージを購入すると，ディスクの中にBACKUP.EXEというハードディスクのバックアップを行うためのツールが入っている．このBACKUP.EXEは単純にコマンドラインから指定したドライブのファイルをディレクトリ情報とともに，順次フロッピーディスクにコピーをする．



コンパイラやユーティリティなど，市販されているものはマスターディスクがしっかり保存されているはずだから，ハードディスクからのバックアップはあまり意味はない．そうなるとバックアップが必要な部分はかぎられてくる．つまりデータや自作のライブラリやソースなどのファイルである．BACKUP.EXEも一応，バックアップするディレクトリやファイルのマスクを指定することができる．

Newton-Backupは，ハードディスクの各ドライブを丸ごとバックアップする機能はもちろんのこと，ディレクトリやファイル単位でバックアップの指定をすることができる．

また，ノート型パソコンのRAMディスクやネットワークの仮想ドライブなど，仮想的なデバイスでもバックアップすることができる．

ワンラインNEWS
アルプス電気は1インチ厚で記憶容量が211Mバイトの3.5インチHDDを開発した．11月からサンプル出荷を開始する．

3つのモードで簡単バックアップ

Newton-Backupのパッケージを開封すると，マニュアル1冊と5インチ2HDと3.5インチ2DDのフロッピーディスク，そしてユーザー登録葉書が出てくる．自分の機種に合わせて，ドライブにフロッピーディスクを挿入する．挿入したドライブがEドライブならMS-DOSのコマンドラインから

```
A>E:INSTALL
```

と入力する．あとは画面の指示にしたがって入力すればインストール作業は完了である．

早速ハードディスクのバックアップ作業を行う．カレントをNewton-Backupをインストールしたディレクトリに移動し，MS-DOSのコマンドラインから

```
A> NTBACKUP
```

と入力する．するとカレントドライブのディレクトリ情報を読みだし，ディレクトリツリーとカーソル位置のディレクトリのファイル内容が表示される．ここで［B］または［f·3］を押せばバックアップの開始である．

Newton-Backupのバックアップには3つのモードがある．「新規」は，バックアップを行うディレクトリやファイルなどの情報を生成する．以後，この情報をもとに作業が行われる．このモードは，はじめてバックアップを行うときに必ず指定する必要がある．「更新」は前回バックアップされたファイルのなかで，更新されたファイルのみを対象にする．「更新，追加」は，新たにバックアップするディレクトリやファイルを指定することができる．今回はもちろん「新規」を選択する．すると，バックアップに必要なフロッピーディスクの枚数があらかじめ表示される．フロッピーディスクを指定したドライブに挿入すればバックアップの開始である．もちろん，未フォーマットのディスクは，フォーマットしながらバックアップしてくれる．次回からは「更新」を選択すればよい．

リストアも万全

リストア作業も基本的にはバックアップと同じである．Newton-Backupを起動し，［R］または［f·4］を押す．どのバックアップをリストアするか，バックアップ時に設定したタイトルの一覧を表示するので，選択する．

リストアは，バックアップしてあるすべてのファイルを復元することはもちろん，リストアするファイルを選択することもできる．その際にもバックアップしてある日付の古いファイルで新しいファイルを上書きすることはない．

当然，フロッピーディスクもハードディスク同様に読めなくなる可能性をもっている．バックアップディスクがすべて完全な状態でないとリストアできないのであれば，バックアップの信頼性に欠ける．その点Newton-Backupは，バックアップディスクの一部に不良セクタが発生しても，その部分のファイル以外はすべてリストアが可能である．極端にいえば，バックアップしたディスクが数枚なくなっても，残っているファイルはすべてバックアップが可能ということになる．

最近はあらかじめMS-DOSフォーマットされたフロッピーディスクが販売されている．このディスクを使用すれば，専用フォーマットを採用したバックアップツールのように，フォーマット作業に時間を費やすこともないので素早いバックアップが可能になる．

Newton-Backupは，むずかしいDOSの知識を必要とせず，強力なヘルプ機能と合わせて，同社のファイル管理ツール「Newton-98」同様のユーザーインターフェースを採用しているので，画面を見ながらコマンドを選んでいくだけで簡単にバックアップがとれる．やはり，ハードディスクにエラーが出てから後悔するよりも，日頃のバックアップで安心を買いたいものである．

■Newton-Backup
販売：（株）アドミラルシステム（TEL 0482-52-4646）／価格：9,700円（消費税含まず）／適応機種：日本電気PC-9801シリーズ，エプソンPC-286/386シリーズ（ノーマルレゾリューションのみ対応）／動作環境：MS-DOS Ver.2.11以降（一部の初期バージョンを除く）／必要メモリ：640KB／供給メディア：5インチ2HD，3.5インチ2DD同梱

オンラインソフト採集帳 12

圧縮ファイルを扱うためのユーティリティ LHQ

ひとことで圧縮ファイルの復元といってもLZH, ZIP, ARC, PAKなど，拡張子によって復元ツールを使い分ける必要がある．しかも，引数の渡し方もまちまちで，タイプの違う圧縮ファイルをいくつも扱うときはなかなかたいへんである．LHQはそんな悩みを解消してくれるツールだ．

準備と起動

LHQは各種圧縮ツールを利用するためのユーティリティであり，LHQ自身が圧縮や復元を行うわけではない．LHA.EXEなどのファイル圧縮ツールやページャーやエディタなど，テキストファイルを参照するためのプログラムをあらかじめパスの通ったディレクトリに準備する必要がある．各ユーティリティの準備が整ったら，次は環境変数の設定である．MS-DOSのコマンドラインから

```
A>SET TMP=H:¥…………①
A>SET PAGER=MIFES……②
```

などと設定する．①の環境変数はテンポラリドライブの設定で，RAMディスクなど高速なアクセスができるドライブが設定されているときには有効となる．②は圧縮されたファイルの中にあるテキストファイルを参照したいときに利用するプログラムを指定する．デフォルトではMIELが設定されている．準備ができたらLHQを起動する．MS-DOSのコマンドラインから

```
A>LHQ
```

と入力する．起動パラメータにはパス名が指定できる．

LHQのコマンド

ファイルの選択

各コマンドはカーソル位置のファイルに対して実行される．複数のファイルに対して実行したいときは[SPACE]キーでマークすることにより，複数のファイルを選択することができる．

ファイルの復元

[E]キーを押すと，圧縮ファイル内のすべてのファイルが復元される．復元するディレクトリを指定することができる．復元に使うツールはLHQ自身がファイルの拡張子から判断する．

ファイルの圧縮

[A]キーを押すと，マークしたファイルまたはカーソル位置のファイルを圧縮することができる．また[M]キーなら移動である．圧縮に使用するツールは[INS]また[DEL]キーで変更することができる．選択できる圧縮ツールはLHA以外にもPKZIPやPAKなどを選択することができる．ほかにも，現在表示しているディレクトリのファイルを含む，サブディレクトリにあるファイルをすべて圧縮する[B]や[R]（移動）もある．

ファイルの内容表示

[RETURN]キーを押すと，圧縮されたファイルの一覧を見ることができる．さらに圧縮されたファイル内で，カーソル位置のファイルの表示[RETURN]，実行[X]，編集[T]，削除[D]，検索[F]，復元[E]を行うことができる．

コメントモード

[HELP]キーを押すとコメントモードとなる．このモードでは，1画面に表示できるファイルの数は少なくなるが，圧縮ファイル内も含めて各ファイルに対しコメントを付加することができるので，オンラインソフトの管理などには効力を発揮するのではないだろうか．

ほかにも差分ファイル，自己復元型圧縮ファイル，DIETやPKLITEによる圧縮実行ファイルの生成や復元なども行うことができる．

FDと組み合せて使う

ファイル管理ユーティリティFDを使えば，拡張子の違いを気にすることなく圧縮ファイルを復元することができる．ただし，FDでは圧縮ファイル中の任意のファイルだけを取り出すことができない．環境変数「EXARC」を設定すれば，FDで圧縮ファイル上にカーソルがあるときに[SHIFT]+[RETURN]を押せばLHQが起動し，圧縮されたファイルのリストが表示されて任意のファイルの復元や削除が可能となる．また，環境変数「AXARC」を設定すれば，MIELのファイル選択画面でも同様のことが可能となる．

新しい圧縮ツールにも対応を

圧縮効率ではLHA，圧縮・復元速度でPKZIPが優秀だといわれてきたが，最近ではLHAと同等の圧縮率をほこりPKZIP同等の速度をもつARJがドイツから上陸してきた．MacintoshのStuff itで圧縮したファイルを復元するツールもある．ときどき「ZOO」と

ワンラインNEWS
日本電気は「A-VX」に加え「日本語MS-DOS」の採用により，PC-9800シリーズのソフトウェアを利用することができるオフィスプロセッサ「OP-98」を発売した．

いう拡張子をもつファイルも見かける．ファイル圧縮ツールは，ほかにもたくさんある．すべては無理としても，なるべく多くのツールに対応することを期待したい．

注)

ARJはRobert.K.Jung氏の作成したフリーソフトウェア，DIETはTeddy Matsumoto氏の作成したフリーソフトウェア，LHAは吉崎栄泰氏の作成したフリーソフトウェア，FDはA.Idei氏の作成したフリーソフトウェア，MIELはA.Idei，Paci両氏の作成したフリーソフトウェア，PAKはNoGate社のシェアウェア，PKLITE，PKZIPはPK-Ware社のシェアウェアです．

自己復元型圧縮ファイル：自己復元ファイル・自動復元ファイルとも呼ばれる．復元ルーチンを圧縮ファイル内に付加し，圧縮ファイル自身を起動することで復元を可能にした圧縮ファイル．
圧縮実行ファイル：自己復元型圧縮ファイルに近いが，圧縮されるファイルはEXEやCOMという拡張子をもつ実行可能なファイルである．また復元されるファイルはディスク上ではなくメインメモリ上で，そのまま実行できる．

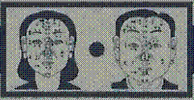
■LHQ Ver.2.11

名称：LHQ／分類：ファイル圧縮・復元ユーティリティ／登録名：LHQ211.LZH／作者：晴田和夫（Qちゃん，NIFTY - Serve:NBA02513/PC - VAN:TAF08339）／動作環境：PC-9801／登録月日：1991年9月23日／復元方法：LHA E LHQ211／入手場所：NIFTY-SERVE（FGALBT 4 #114）

ファイル構成

```
Listing of archive : LHQ211.LZH

  Name            Original    Packed  Ratio   Date      Time    Attr Type  CRC
--------------    --------  --------  ------ --------  -------- ---- ----- ----
  LHQ.EXE            28767     28383  98.6% 91-09-23 02:11:00 a--w -lh5- CF4E
  LHQ211.DOC         10687      4147  38.8% 91-09-23 02:11:00 a--w -lh5- 5878
  README.LHQ           543       353  65.0% 91-09-23 02:11:00 a--w -lh5- 5C1A
  DIRQ.LHQ             155       134  86.5% 91-09-23 02:11:00 a--w -lh5- 03B5
--------------    --------  --------  ------ --------  --------
     4 files         40162     32997  82.2% 91-10-08 20:24:26
```

HEALTHY? ・ UNHEALTHY?



12 目の疲れには……

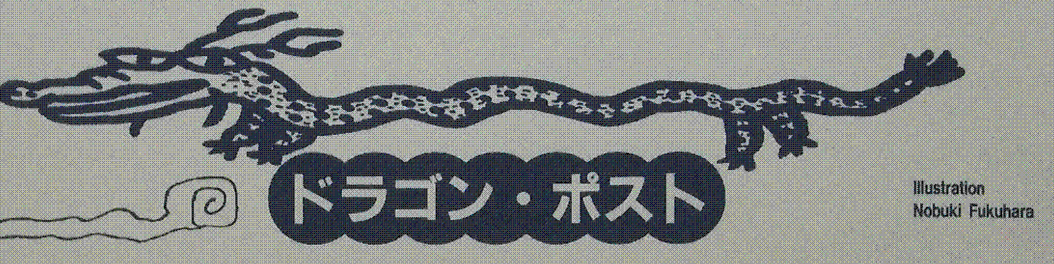
CRTディスプレイで仕事をしていて，目が疲れたという経験は誰にでもあるだろう．目に負担を与えるという点では，長時間本を読めば目が疲れるというのと同じことだが，CRTディスプレイは蛍光灯のように点滅を繰り返しているので，それが目のストレスになるということもいわれている．健康に注意している人は，ディスプレイに保護マスクをかけたり，一定時間ごとに休憩をとったりして，目の負担をやわらげるように心がけているようだ．

考えてみれば，生物としての人間は，もともとは食べ物を得て日々暮らしていけることが最も重要だったわけで，机の前に座って目の前にある紙などを見て仕事をするようにはできていないのだ．だからといって，仕事をやめるわけにはいかないし，急速に視力が落ちることも避けたい……．

それならば，ディスプレイの前で仕事をしているときは，1時間に10分から15分程度は椅子から離れて，気分転換をしたほうがいいようだ．10分以上も気分転換をしていては仕事が遅れてしまうというのなら，デスクを離れて少し歩くだけでもかなり効果はあるという．窓に近くにいる人なら，ときどき遠くを見るのもいい．

食べ物では，ビタミンAを含んでいるものが目にいい．昔から目の病にはヤツメウナギといわれているように，これは15万国際単位（100g中）という，けた外れのビタミンA効力がある．ウナギもビタミンAが豊富な食品として知られるが，5千国際単位だ．しかし，ヤツメウナギは日常生活で普通に食べる食品ではない．手にいれやすく，食べるのが楽な食品を選ぶべきだろう．黄緑色野菜は，ビタミンAが多い．とくに，生ニンジンは4100国際単位も含んでいる．あと，めぼしいものとしてはレバーがある．どちらかというと，好き嫌いのある食品が多いが，ニンジンの生ジュース「確かに効く」といわれている．

松下電器産業はMSXの16ビット規格「MSX TurboR」のパソコン「FS-A1GT」を発売した．別売のマウスを利用することによりGUIを利用したアプリケーションが利用できる．



ドラゴン・ポスト

Illustration
Nobuki Fukuhara

この欄は，読者からのお便りを中心に構成しています．本誌に対するご意見・ご感想などをお待ちしております．鬼より恐い読者からの声ということで，「ドラゴン・ポスト」です．本誌についているアンケートはがきにご意見の欄を設けてありますから，そこに書いてください．ペンネームでも歓迎です．ただし，はがきには本名と住所をしっかり書いてくださいね．掲載誌をお届けします．

★ソフト開発の実態は？

なぜ，プログラム作成は記事になるのに，それに付随するドキュメント作成は記事にならないのであろう．あの苦痛でしかない作業が仕事の大半を占めるという，ソフトの仕事の実態を世に知らしめたい……．(QLQ)

——実態というのはなかなか外からではわかりにくいものです．編集の仕事も何がなくても体力だって知ってました？　QLQさんのように世間に知らしめたい人や知りたい人のために，「プログラマの実態を鋭く（？）えぐる新連載」を企画中ですのでお楽しみに！

★C＋＋ブームにもの申す

最近OOPブームとかで，C＋＋などモテモテのようですが……たいていクラスライブラリが満足になかったり(ブラウザがないのは当たり前)，そんなに「使える」ものなのでしょうか？　Smalltalkなみの環境を標準で使えるようにしてからいばってもらいたい(と思う)．というわけで，私はCが好きです．仕事ではdBASEばかりいじっていますので，フラストレーションがたまります．それはともかく道具としての前提のそろっていない道具(C＋＋）をありがたがる風潮はどこかおかしいのではないですか？（Cと98が好きな「変わり者」）

——OOPというのは，考案されてから数十年にわたる研究段階を経て現在実用化されようとしている技術です．ブームなどとは無関係に学んでおく必要のあるものだと思います．

★マスターするのが先か，バージョンアップが先か

BC＋＋のバージョンアップをしたら，どデカイ箱が送られて来た．うれしいような気もするが置き場所に困ってしまった．インストールしようとすると「最大20MB使用します」とのメッセージが……．ついにソフトもここまできたか．何とかインストールが終わった頃に，また送られて来た修正マニュアルと修正ディスクが追い打ちをかける．マスターするのが先か，バージョンアップが先か．それが問題だ．(頭がパンクしそうなY.Y)

——たしかに最近開発ツールのバージョンアップサイクルが短くなってきているようです．Actorも春ごろ3.0から3.1にバージョンアップしたのに，もう4.0が発売になりました．バージョンアップによる学習曲線についても考える必要があるかもしれませんね．

★100点満点を求めてみたら？

車でもパソコンでも100点満点を求めるものが少なくなり，すべてにおいて80点指向の何ともつまらないものばかりです．貴誌は100点満点を求めてみたら如何ですか(廃刊覚悟で……冗談，ジョーダンですよ．今のままで結構です）（長谷一水）

——やはり製作コストを考えると，さまざまな要求に100％合わせるのは難しいでしょうね．CPUは486で，HDは100MB，あードライブは5インチと3.5インチは1個ずつね……なんて，オーダーできたらいいんでしょうが．

★締切に間に合わない！

近頃全然Programmingが進まない．どうしよう．このままでは締切に間に合わない．締切といっても，自分で勝手に決めたものだから，無視してもよいのだけど．原因はきっと，Bikeと英文で書かれたReferenceのせいだ．ところで，みなさんはどうやってProgramを制作してますか？　何か期日のようなものを設けないと，ズルズルと制作期間が長びいてしまうような気がするのですが．(近頃Bikeと英文の和訳のことしか頭に置いとけない　狂夢)

——たしかに締切がないとズルズルしてしまうでしょうが，自分で締切を作るなんて……エライ！　ある人が，プログラミングしてる時は，まるで竜宮城にいるようだといっていました．「月日のたつのも夢のうち……」

★長雨が心配

AppleとIBMどちらも目が離せませんね．DOSだけでなく，Mac OSにも目を向けて下さい．ところで長雨からくる湿気はパソコンやFDにどんな影響を与えるかと心配になりますが，編集部の皆様もお身体には気を付けて下さい．(みづたま)

——ハイ，今年の台風＋秋雨前線はすごかったですね．編集部ではプリンタマシンに98のE(！)を使っているのですが，これが雨の日にはとたんに機嫌が悪くなってた

いへん．意外とデリケート(？)なんだから……．

★思ってもみなかった問題が……

最近はどんな会社にも，コンピュータに関係するシステムが導入されていると思います．我社も精算システム(単なるお金の計算)を導入しました．ところが，私(37才)が使用前には想像もしなかったのですが，プリンタの字が小さすぎるとの事．(ちなみに主に使用する人の年齢平均62才)(佐藤正憲)

――簡単に印字サイズを大きくできれば問題もないのでしょうが，そこはケースバイケース．拡大コピーしてさしあげるしかないのでは……．

★ワタクシの人生は……

ワタクシの人生　(始)まっとうな営業マン→BASICで遊ぶ→コンピューター室のCOBOLプログラマに転職→システム設計屋→UNIXマシンでCTSデータ処理→NetWareでDBソフト作り……．うーむ．ワタクシはどこへ流れてゆくのであろーか．※ひとつの会社の中です．(ライク・ア・ローリングストーン)

――同じ会社の中でこれだけ転職できるなんて，柔軟性がある証拠でしょう．流れていく先が自分の望む方向ならラッキーですよね．

★486マシンを買った！

PC-H98S8を買ってしまった．やっぱり486SXは速い！e^2GCの威力でWINDOWSの描画も一瞬．ボーランドC++にバージョンアップした．プリコンパイルヘッダとターボドライヴ(DOSエクステンダ)で，コンパイルの速さにビックリ．次はNESAバスのＥバスのバースト転送で速さに驚きたい……でも，速すぎてゲームにならないものもある．速いマシンにするたびに，手持ちのゲームを全部やり直してしまう私……マシンが泣いとる．(加藤克哉)

――ビックリするほど速いなんてうらやましい！　手持ちのゲームを全部試してみてしまう，その気持ちも妙にわかってしまう．

★周囲の注目が……

毎月，私の机上に配送されてくるP.P.の表紙に，周囲の方々が注目しはじめました．はじめは「なんだ，こりゃ」といった反応でしたが，今ではみんな楽しみにしているようです．特にトナリの席の人間がそっくりなものですから……．(パタリロ)

――実は編集部にも表紙の人物に似た人がおります．結構，いがちな人物なんでしょうか．これからもぜひP.P.を見せびらかしてください．

★世の中はおもしろい

10月号のパソコン博物館を読んで……当時，私の友人がMZを持っていた．私は彼に「パソコンで何するの」と聞いた．彼は，返事に困り「とりあえずアドレス帳を作った」と答えた．そこで私は言った「俺の住所録はポケットに入るもので，すぐにわかるよ」……彼は答えなかった．その後何年かして，私はMSXを買った．MSXは半年でアキで9801に買い替え，彼は当然のごとくX68Kに移った．私はMSXを“おい”にあげた．“おい”はその後パソコンに熱中し，高卒と同時にコンピュータ学校に入学したそうな……世の中は，いつまでも面白いものです．(桜木泰教)

――甥子さんもおじさんからMSXを貰わなければ……もっと別の方向へ進んでいたかもしれませんね．でも，それがきっかけで好きなものが見つかったのですから，きっとおじさんに感謝(？)していることでしょう．

★パソコン導入の犠牲は

職場へは未だにパソコンが導入されていません．毎年予算要求をして事務合理化の計画を出すのですが……もっとも“生首を切る”案をあげていないので，実現しないのかも?? 機械と人間をテンビンにかけられますか？導入した上で，マイルドな方法であればどうにかなるのですが，導入と同時に“チョン”ではネ……．皆さんどうですか？　私が甘いのかな．(合理化太郎)

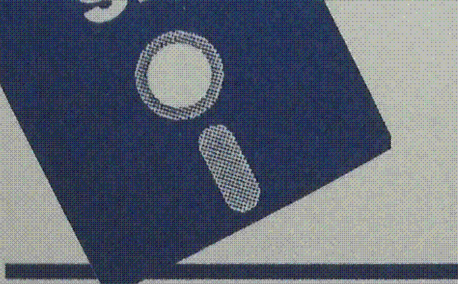
――機械でできることなら機械化されるべきだと思います．たとえば駅の自動改札とか……．でもパソコンは，人が使うものであって，人の代わりになるものではありませんから，「パソコン導入＝人減らし」となるのは，ちょっと変だと思います．ルーチンワークをパソコンに肩代わりさせて，その分もっと別の仕事をこなしてこそ，パソコン導入の意味があると思うのですが……．

★苦闘の14時間

PC-386GS5を購入してから，しばらくノーマルモードで使用していたが，このたびようやくハイレゾ対応モニターを購入した．ところが起動してみたらハイレゾモードではハードディスクは認識しないわ，増設メモリは認識しないわ……マトモに動くまで14時間かかった(久々の徹夜)．それにしてもマニュアルが悪い．すべてのね……．(高橋健一)

――何分冊にもなる分厚いマニュアルでは，どこに何が書いてあるのかを探すだけでも一苦労．まして何が原因なのかわからない時には，あっちのマニュアル，こっちのマニュアルと……14時間とはホントご苦労様でした．

(敬称略)



付録ディスクのご案内

このフロッピーディスクに含まれる実行ファイル，ソースリストおよびテキストファイルは「月刊Programmer's Page」1991年12月号に掲載されたものです．

付録のフロッピーディスクは，5"2HD（8セクタ／トラックMS-DOS）でフォーマットされています．3.5"や2DDなどへのコンバートサービスは行っておりません．なお，オンラインサービスSE-NET（03-5276-1297）でも，同じ内容のファイルを用意しています．

フロッピーディスクの内容は次の通りです．本誌およびディスクに含まれるドキュメントファイルをよく読んでからご利用ください．

■ディスクのおもな内容

●付録ディスクについての説明
ディレクトリ名：ルート
収録ファイル名：
README.BAT（ディスク内容の説明を読むためのバッチファイル，PC-9801専用）
README.DOC（ディスク内容の説明，MS-DOSテキストファイル）
ONLINE.BAT（オンラインソフトの説明を読むためのバッチファイル，PC-9801専用）
ONLINE.DOC（収録されているオンラインソフトの簡単な説明）
V.EXE（ミニページャー，PC-9801専用）
BOOK.TXT（翔泳社書籍目録）

●特集「やっぱり僕らはBASIC！」
①「データ構造＋プログラム構造＝ニューベーシック」（p.2）
ディレクトリ名：TOKU
収録ファイル名：TOKU1.LZH
②「構造化プログラミングを学ぶ」（p.11）
ディレクトリ名：TOKU
収録ファイル名：TOKU2.LZH
③「Microsoft BASIC Ver.7.1」（p.16）
ディレクトリ名：TOKU
収録ファイル名：TOKU3.LZH
④「Microsoft VisualBasic」（p.25）
ディレクトリ名：TOKU
収録ファイル名：TOKU4.LZH
⑤「BASIC/98」（p.40）
ディレクトリ名：TOKU
収録ファイル名：TOKU5.LZH
⑥「TrueBASIC」（p.54）
ディレクトリ名：TOKU
収録ファイル名：TOKU6.LZH

●プログラム『遊』工房（p.74）
「JOKE/TOKEL」プログラム
ディレクトリ名：PLAY
収録ファイル名：PLAY.LZH

●MS-DOSツール・ラボ（p.80）
フリーメモリ残量調整プログラム
ディレクトリ名：TOOL
収録ファイル名：TOOL.LZH

●実践C塾（p.94）
ちょっと便利な関数群
ディレクトリ名：JUKU
収録ファイル名：JUKU.LZH

●98活用アセンブラ技法（p.101）
時間取得・表示プログラム／プリンタスプーラの作成
ディレクトリ名：98ASM
収録ファイル名：98ASM.LZH

●綴じ込み付録
ディレクトリ名：DATA
収録ファイル名：QBFUNC.TXT（MS-DOSテキストファイル）

●オンラインソフト採集帳（p.140）
本誌今月号のオンラインソフト採集帳で紹介した各種書庫管理ツールを活用するユーティリティ「LHQ」が収録されています．また「.LZH」という拡張子を持つ圧縮ファイルを復元するのに必要な，高圧縮復元書庫管理ソフト「LHA」も収録されています．
ディレクトリ名：ONLINE
収録ファイル名：
LHQ211.LZH（LHQ Ver.2.11）
LHA213.EXE（LHA Ver.2.13，MS-DOS汎用）

●クイズ・ランダムスクリーン（p.142）
クイズ解読ツール「PUZLE」
本誌掲載のクイズ・ランダムスクリーンの画像をディスプレイ上に再現し，マウスを使って並べ換えることにより，解答を得ることができるユーティリティです．PC-9801シリーズ専用です，詳しい使い方はPUZLE.DOCを参照してください．
ディレクトリ名：OMAKE
収録ファイル名：
PUZLE.EXE（解読ツール，PC-9801専用）
PUZLE.DOC（PUZLEの説明）
CONFIG.SYS（PUZLE用CONFIG.SYS）
AUTOEXEC.BAT（PUZLE用AUTOEXEC.BAT）
RNDSCREN.LZH（画像データ）
CHURCH.LZH（11月号の解答データ）
CHURCH.BMP（Windows 3.0の壁紙）

※使用上の注意
本誌に掲載されたプログラムおよびデータを利用するには，MS-DOSの基礎知識が必要です．このディスクには，MS-DOSのシステムは入っていません．別途，MS-DOSが必要となります．また，ソースファイルのコンパイルなど，別途コンパイラやユーティリティが必要な場合があります．

■圧縮ファイルについて

ディスクスペースを有効に使うために，多くのファイルは圧縮した上で収録しています．「.LZH」という拡張子のついたファイルは，高圧縮書庫管理ユーティリティ「LHA」によって圧縮されたファイルです．これらのファイルを利用するには，まずLHAで復元する必要があります．LHAの利用方法については，ONLINE.DOCをご覧ください．

バーチャルスペースガイド

P子　某ソフトハウスで企画営業を担当している．26歳．
C平　某メーカーのシステムソフトウェア開発部門に所属．25歳．

P子　コンピュータウイルス騒ぎも一段落しているみたいね．一時は，一般紙がトンマな記事をよく載せていたけど．

C平　ウイルスという名前でイメージが先行してしまって，好奇の目でみられたことは確かだろうね．バグや操作ミスまでがウイルスのせいにされたりしちゃって．

P子　でも，最近週刊誌に「エイズ爆発のXデー」という記事が載ったように，深く静かに潜行して広がっているということもあるんじゃない．

C平　うん，マスコミとは関係なく，まじめなウイルス対策がネットワーカーの間ですすんでいるみたいだよ．

P子　ウイルスとは違うけど，ハッカーもののドキュメンタリーが本になったでしょ．なんていたっけ……．

C平　「カッコウはコンピュータに卵を産む」．

P子　あ，それそれ．でも変な題名．

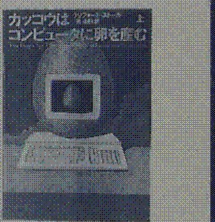
C平　カッコウはほかの鳥の巣に侵入して自分の卵を産んで，育てさせるんだ．雛のほうが育ての親より大きかったりして，ちょっとかわいそうなんだけど．ハッカーは卵にあたるプログラムを作って，システムがこれに「特権」という餌を与えるように擬装するわけ．

P子　なーるほど．

C平　ドイツのハッカーがアメリカの研究所や軍事用のネットワークに侵入する事件があったんだけど，実際にそれを追跡した研究者が書いたドキュメントなんだ．いま，ミステリー小説以上に面白いって評判．

■カッコウはコンピュータに卵を産む

クリフォード・ストール著／池央耿訳
草思社　上・下巻各1,900円

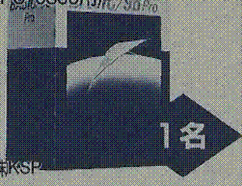


天文学研究のかたわら，新米のシステム管理者となった著者が見つけた，75セントの使用料金不足，原因を究明するうち，システムに侵入する謎のユーザーが浮かび上がる．1年以上に及ぶ孤軍奮闘のハッカー追跡のドキュメント．バークレーの研究者の生活ぶりやコンピュータネットワークの現状なども描かれており，読み物としても十分おもしろい．

読者プレゼント

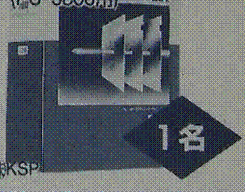
応募は本誌に添付のアンケートはがきをご利用ください．(1991年12月17日消印有効)．なお，応募者多数の場合は，抽選になります．当選者発表は発送をもって代えさせていただきます．

1 BASIC/98Pro
(PC-9800用)



1名
提供：㈱KSP

2 BASIC/98Fast
(PC-9800用)

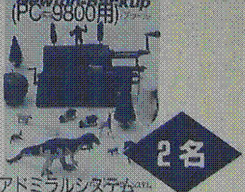


1名
提供：㈱KSP

3 Microstft BASIC Ver.7.1
(PC-9800用)

1名
提供：マイクロソフト㈱

4 Newton-Backup
(PC-9800用)

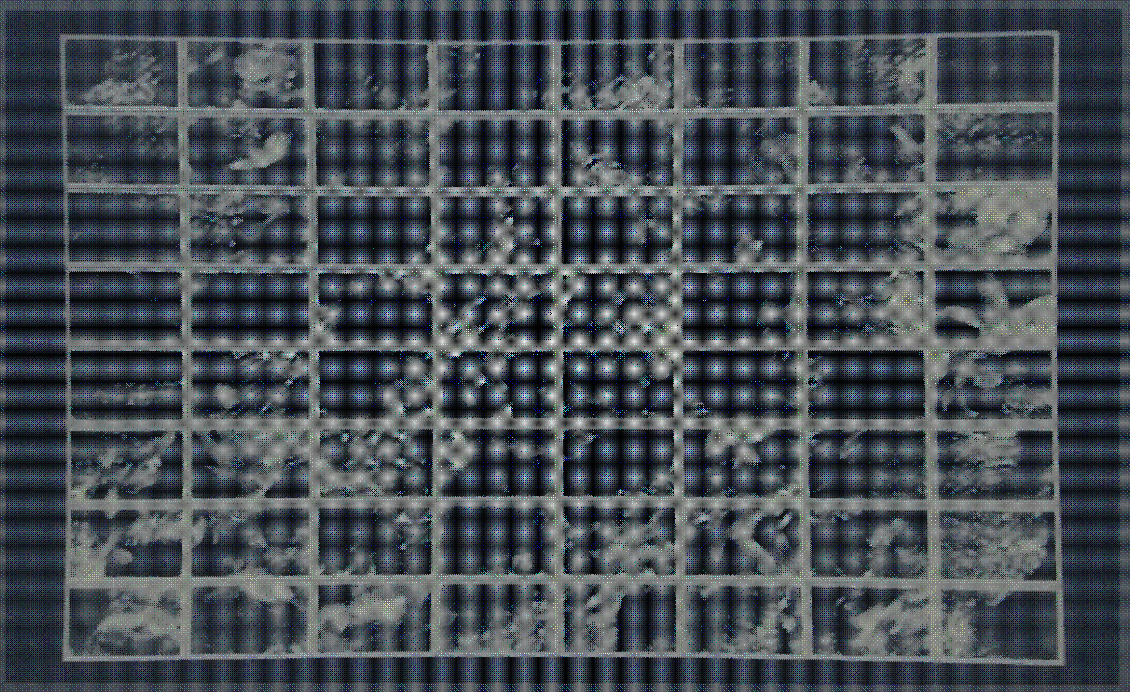


2名
提供：㈱アドミラルシステム

クイズ・ランダムスクリーン

ある１枚の画像を64分割し，ランダムに並べ替えてあります．
さて，もとの画像は何を写したものでしょうか．

ヒント：今月は簡単なのでノーヒント

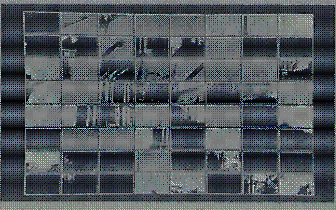


正解者の中から30名様に本誌表紙イラスト（日暮修一画）入り月刊プログラマーズ・ページ特製オリジナルマグカップをさしあげます．
応募方法：本誌に添付のアンケートはがきをご利用ください．
締め切り：1991年12月17日（消印有効）
なお，当選者の発表は発送をもって代えさせていただきます．

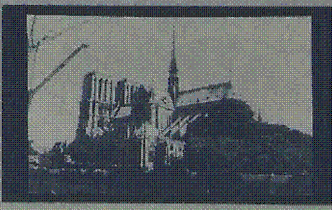
＊画像データは，付録ディスクやBBSでもサービスしています．

11月号の問題と正解

【問題】



【正解：ノートルダム寺院】



SE-NETからのお知らせ

SE-NETでは開設以来，オンラインサービスを行ってきましたが，回線混雑のため非常に接続しにくい等の問題が出てまいりましたので，GUESTによるダウンロードは中止させていただきます．今後は読者のみなさまと編集部とのコミュニケーションの場としてご利用ください．

●BBSへの接続

SE-NET　03-5276-1297

通信速度	300/1200/2400bps
接続プロトコル	CCITT V.22bis/V.22/V.21
エラー訂正プロトコル	CCITT V.42，MNPクラス５
漢字コード	シフトJIS

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

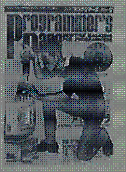
月刊プログラマーズ・ページ バックナンバーのご案内

■定価・各980円

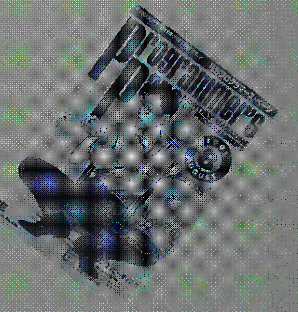
（1991年1月号「特集・プログラマの仕事場」・2月号「特集・Windowsの時代である!」・3月号「特集・この言語を使いなさい!!」・4月号「特集・C読本」は品切れです）

1991年5月号 5インチFD付き

特集・基礎力としてのアセンブラ



プリミティブなプログラミング言語でありながら，それ故に無視することができないアセンブラ。コンピュータを理解し，その能力をひきだすためにもう一度アセンブラをみなおす。



1991年6月号 5インチFD付き

特集・研究「プログラミング統合環境」

今流行のエディタ，コンパイラ，リンカ，デバッガを統合した開発環境について，その歴史的系譜から最新のシステムまで徹底的に検証する。

1991年7月号 5インチFD付き

特集・最新「デバイスドライバ」研究

パソコンの拡張性を支えるソフトウェアインターフェースであるデバイスドライバ作成の技法と特徴をMS-DOS，Windows，OS/2の各環境ごとに解説する。

1991年8月号 5インチFD付き

特集・はじめてのオブジェクト指向プログラミング

90年代のプログラミングパラダイムとして注目される"オブジェクト指向"の歴史的な発展の過程から基礎概念，分析・設計の方法までを網羅的に解説する。

1991年9月号 5インチFD付き

特集・グラフィックスプログラミングの世界

グラフィックスプログラミングの基礎からフレームバッファの活用，レイトレーシング，Windows，IBM PCのグラフィックスなど，その技法を徹底的に解説する。

1991年10月号 5インチFD付き

特集・286/386メモリ活用プログラミング

バンク式メモリ，EMS，XMSからプロテクトモードの活用まで，MS-DOSがもつ640KBの壁を破るための数多くのプログラミングテクニックを解説する。

1991年11月号 5インチFD付き

特集・Windowsプログラムの作り方

正統派MS-C＋SDKをはじめ，Visual-Basic，BC＋＋，TPWなどの新しい開発環境，ツールによるWindowsプログラムの作り方を探る。

「月刊プログラマーズ・ページ」のバックナンバーは，ご注文いただければ，全国どこの書店でもお求めになれます(書店に届くまで1か月ほどかかります)。小社に直接ご注文なさるときは，何月号かを明記の上，現金書留にて雑誌価格＋送料を右記までお送りください。送料は1部61円です（ただし7月号は1部66円）。2冊以上の場合は，号数・冊数により送料が異なりますので，小社までお問い合わせください。

〒102 東京都千代田区平河町
2-4-14 平河町KSビル4階
(株) 翔泳社
マーケティング部PP係宛
TEL 03-3263-0447

Programmer's Page

MONTHLY MAGAZINE FOR PROGRAMMERS

月刊プログラマーズ・ページ1月号

(12月18日発売)

付録：5″ 2HDフロッピーディスク付き!!

特集

プログラミングツール大集合

そのプロフィールと使いこなしのコツ

プログラマは，プログラムの開発・メンテナンスの各フェーズでさまざまなプログラミングツールを使いこなさなければならない．さらに，優れたツールを使用することによってプログラミングのクオリティを上げることも可能になる．これらのツールについて，その利用法から使いこなしのコツまでを紹介する．

月刊プログラマーズ・ページ　1991年12月号
1991年12月1日発行

発行所　株式会社翔泳社
東京都千代田区平河町2-4-14平河町KSビル4F・5F
TEL 03-3263-0447　〒102
発行人　長廻健太郎
編集長　清水　隆
副編集長　鈴木嘉平
編集スタッフ　杉野貴美子＋上田純美礼＋芳山充弘
デザイン　SE（翔泳社デザイン研究所）
広告スタッフ　相澤正夫＋澤本良知＋種村嘉彦
協力　根本裕三郎＋(株)ドモン・マインズ＋佐藤裕美＋小林弘人(図版デザイン)
写植　株式会社東巧社＋株式会社トーヨー・プリント社
印刷　凸版印刷株式会社
定価　980円（本体951円，税29円）

© 1991 SHOEISHA
本書の一部または全部を著作権法の定める範囲をこえ，無断で複写，転載，テープ化，データファイル化することを禁じます．
＊本書の内容に関するご質問は，ご面倒でも必ず文書にて編集部までお問い合わせください．
＊本誌に記載した会社名，ハードウェア・ソフトウェアの商品名は，各社の商標または登録商標です．

EDITOR'S PAGE

●秋は，イベントと新製品発表の連続だ．春と秋に集中するのが年中行事のようになってしまった．変化の激しい業界ではあるが，1年前とくらべて製品の状況は大きく変わったような変わらないような……．「あのメーカーがいつあれを出すか」というレベルでは確かに変わっているが，1年前に予測もできなかった製品はない．技術開発，市場環境などさまざまな要素の上に製品が登場してくる事情もある．また，突拍子もないものが飛び出す余地がないほど業界が成熟してきているのだろうか．(TA)

●BASICといって思い出すのは十数年前，高校のクラブでセッセとFORTRANでプログラムを作っていた頃，後輩の1人がワンボードマイコンを買った．当時の田舎の高校生にとって憧れの的だったワンボードマイコン，値段が高すぎてとても私には買えなかった．嬉しそうにBASICでプログラミングしている後輩を横目に見ながら，「フンッ／ BASICなんかよりFORTRANのほうがずっといいわい．」とつぶやいた……．あの日以来，私はBASICか嫌いである．(嘉平)

●はやいものでもう12号．あッという間に今年もおわりですね．年頭にたてた計画も，ほとんど実現できなかったな……．たいした計画でもないのに実現できないのは，意志の弱さか体力のなさか，ということで来年は……．いえ，やめておきましょう．きっとまた同じことになるんだから．30年生きてきてようやく気づいた私．(ま)

●近況報告．いつのまにか白鳥の渡ってくる季節になったことを，朝のワイドショーの瓢湖からの生中継を見て知った．気がつけば街にはツバメの姿はなく，伊良湖の渡り鳥の季節もとうに過ぎていた．きっとこの号が発売されるころには，皇居のお堀もシベリア帰りのキンクロハジロやらヒドリガモやらオナガガモなんかでいっぱいだろう．愛らしいジョウビタキにもまた会えるだろうな，なんて考えながら秋晴れの空の下を散歩していたら，アスファルトの上になぜかカマキリが一匹．東京の自然もまだまだ捨てたもんじゃないです．(kos)

●米国のパトカーにABSのついた車を採用したところ，制動距離が伸びて危険だと警察官から苦情が相次いでいると，某カー雑誌に書いてあった．フルブレーキを掛け，白煙を上げながら止まったほうが制動距離が短くなると真剣に思っているあたり，洋画のワンシーンもまんざら冗談じゃないと思えてくる．フルアクセル・フルブレーキで行うシグナルグランプリとチキンレースも，米国では迫力だけのようだ（笑）．(Yoshi)

お詫び

●連載「佐々木哲のニューヨークシステム通信」は，著者の都合により休載いたしました。

愛読者カードご記入のお願い

本誌をお買い上げいただき、誠にありがとうございます。
お手数ですが、今後の誌面づくりの参考にさせていただきたいと存じますので、
各欄にご記入の上、ご返送くださるようお願い申しあげます。

月刊プログラマーズ・ページ READER'S SURVERY CARD 愛読者カード

本誌の号数　　年　　月号

本誌を知った理由
☐1. 書店店頭　☐2.他社出版物（　　　　）
☐3.小社出版物　☐4. DM　☐5.口コミ

パソコンの所有形態（機種名）
☐1. 個人所有　（　　　　）
☐2. 職場にある（　　　　）
☐3. 持っていない

本誌を購入した書店　　市（区）　　書店

よく使用するソフトウェア

よく使用するOS

今月号でおもしろかった記事（いくつでも）

今後取り上げてもらいたい特集のテーマ

貴方が選ぶベスト３

パソコン雑誌	①	②	③
ソフトウェア	①	②	③
パソコン本体	①	②	③

本誌についてのご感想、お気づきの点、その他どのようなことでも結構ですので、ご意見をお聞かせください（「ドラゴン・ポスト＝読者の声」欄に掲載させていただく場合もあります。ペンネームをご希望の方はあわせてご記入ください）。

月刊プログラマーズ・ページ 広告資料請求カード

8	会社（学校）名		9	所属（学部）		10	従業員数	

※下記の該当する番号をご記入ください。

11	業種		12	職種		13	パソコン利用状況	

業種　①製造業　②金融・保険・不動産業　③商社・卸・小売業
④建設業　⑤通信・運輸　⑥システム・ソフト
⑦その他サービス業　⑧公務員　⑨その他

職種　①コンピュータ（ハード）技術者　②コンピュータ（ソフト）技術者　③エレクトロニクス技術者
④プログラマ　⑤その他の技術者　⑥専門職
⑦事務・営業職　⑧経営管理職　⑨商工サービス
⑩教職　⑪学生　⑫その他

パソコン利用状況　①会社で利用　②個人で利用　③会社でも個人でも利用　④利用していない

●希望される広告資料　　年　　月号

広告資料請求No. / 希望項目												
A 購入希望	A	A	A	A	A	A	A	A	A	A	A	A
B 説明希望	B	B	B	B	B	B	B	B	B	B	B	B
C 資料希望	C	C	C	C	C	C	C	C	C	C	C	C

定期購読のご案内

綴じ込みの振替用紙を使って、本誌の定期購読をお申し込みになれます。
1年間12冊＝11,760円(本体11,417円/税343円)
郵送料サービス

郵便はがき

料金受取人払
麹町局承認
3908

差出有効期間
平成4年11月
30日まで

切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
マーケティング部 行

月刊プログラマーズ・ページ
READER'S SURVEY CARD　愛読者カード

氏名	性別 □1男 □2女	生年月日 19　年　月　日
住所　〒		電話 (　　)　－

職業
□1ソフト技術者 □2ハード技術者 □3技術系会社員
□4事務系会社員 □5営業系会社員 □6教師
□7一般公務員 □8経営管理職 □9理系学生
□10文系学生 □11その他(　　　)

ユーザー区分
□1プロ
□2マニア
□3一般ユーザー

プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

郵便はがき

料金受取人払
麹町局承認
3910

差出有効期間
平成4年11月
30日まで

切手不要

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係行

＊当社で記入

1	SE	2	受付No.	

月刊プログラマーズ・ページ広告資料請求カード

3	会員番号	

＊すでに会員番号をお持ちの方はご記入ください。

4	フリガナ 氏名	姓　　　名	5 生年月日	19　年　月　日
6	自宅住所	〒　電話 (　　)		
7	会社住所	〒　電話 (　　)		

資料送付希望先の番号に○印をつけてください。

払込票

通常払込料金加入者負担

口座番号	東京 4	154493
加入者名	株式会社 翔泳社	
金額	11760	
払込人住所氏名	※	
備考		
受付局日附印		

切り取らないで郵便局にお出しください。

記載事項を訂正した場合は、その箇所に訂正印を押してください。

払込通知票

通常払込料金加入者負担

口座番号	東京 4	154493
金額	11760	
料金	払込み	特殊 円
備考		
受付局日附印		
加入者名	株式会社 翔泳社	
払込人住所氏名	※（郵便番号　　） TEL（　　　）	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

各票の※印欄は、払込人において記載してください。

算株式会社

たのは1962年。日本の情報
史と実績を誇る企業のひとつ
報処理サービス会社として設
ライン化などで実績を築いて
いうイメージが強いが、最近
して多様な分野での展開が目

訪ね、SEの山田顕義氏にプ
EWS関連のソフトウェア開
に、人材開発部部長の浅田恒
の考えをうかがった。

私自身はパソコンやEWSの可能性に対して興味をもって見ていましたし、パッケージソフトなど自分が企画から関わって作りあげていくソフトづくりをしてみたいとも思っていました。それで別の会社に移ることを考えはじめたわけです。それと私は新潟出身なので、できれば地元で仕事のできる会社に、とも考えていました。

転職した当初は、新潟支店でオフコンのシステム関係の仕事をおもにしていましたが、パソコンやEWSのソフ

でさまざまな業種・企業向けのシステム開発にあたっている。

「現在企画している仕事としては、まずシルバー産業向けのものがひとつ。これはパソコン数台をNetWareを使用してLANで結んだグループウェアを作ろうというものです。現在シルバー産業は非常に脚光を浴びていますが、一方で社会的な責任も重い分野ですから、厚生省の指導も強化されつつあり、最近は経営的にきちんと数字を出さないと商売ができなくなってきたんです

そこで、その辺りの市場性を狙っ
コンサルティング会社とタイアッ
てシステムを作ろうということで
パッケージというところまではい
いませんが、10本、20本という数
っていこうと考えています。
れから、アップルのマッキントッ
用のシステムにも取り組んでいま
いま進めているのは航空会社の経
貨物の運送実績などを管理するシ
ムなどですね。
た、当社ではソニーのNEWSを使
UNIXのシステムを以前から手
てきたんですが、私自身もNEWS
以前から興味を持っていましたの
航空会社向けのNEWSのソフトも
しているんですよ。これは当社でデータ処理をしている毎月何千ページにもおよぶ航空会社の帳票類をもっと見やすいかたちに加工しようということで、データをCD−ROMに落とし、これをNEWS上で加工して出力するというソフトをいま作っているところです。

NEWSからPC98そしてマッキントッシュまでいろいろでしょ。いまはそれぞれのソフトの用途やマシンの特長を生かしたソフトづくりに挑戦しながら、積極的な姿勢で取り組むよう心がけています。」

定期購読のご案内

■定期購読は、1年間12冊と限定させていただきます。

■購読料のお支払は、この振替用紙をご利用ください。銀行振込、現金書留によるご送金は、ご遠慮ください。

■振替用紙に、ご住所、お名前をご記入のうえ、郵便局の貯金窓口にてお支払ください。

■小社より領収書の発行はいたしませんので、郵便局にて「郵便振替払込金受領証」をお受取ください。

■お申し込みいただいた方には、弊社より直接発送いたしますので、到着が発売日より遅れることがございますが、ご了承ください。

月刊プログラマーズ・ページ　定期購読直接申し込み書

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

年　　月号より

年間購読料 11,760 円(税・送料込み)

■ご希望の購読開始月号をご記入ください。

この欄は、加入者あての通信にお使いください。

■送付先（住所・氏名は正確にご記入ください）

住所	□□□-□□	フリガナ	
		氏名	
		TEL	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。　（郵　政　省）

郵便はがき

102-00

東京都千代田区平河町2-4-14
平河町KSビル

株式会社 翔泳社
月刊プログラマーズ・ページ
広告資料請求係行

2 受付No.

マーズ・ページ広告資料請求カード

＊すでに会員番号をお持ちの方はご記入ください。

名

5 生年月日　19　年　月　日

□7 一般事務員　□8経営管理職　□9理系学生　□3 一般ユーザー
□10文系学生　□11その他(　　　　　)

プレゼント記入欄	クイズ・ランダムスクリーン回答欄
希望商品番号	答

6	自宅住所	〒　　電話　(　　)	資料送付希望先の番号に○印をつけてください。
7	会社住所	〒　　電話　(　　)	

綴じ込みの
1

Programmer's Page MONTHLY MAGAZINE FOR PROGRAMMERS

広告企画

ちょっと気になるこの会社

日本電子計算株式会社

日本電子計算㈱が設立されたのは1962年。日本の情報処理・ソフトウェア業界では歴史と実績を誇る企業のひとつである。金融・証券関係の情報処理サービス会社として設立され、証券システムのオンライン化などで実績を築いてきた同社は"地味で堅実"というイメージが強いが、最近はシステムインテグレータとして多様な分野での展開が目立ってきた。

東京・江東区の本店事業部を訪ね、SEの山田顕義氏にプログラマの現場からパソコン・EWS関連のソフトウェア開発を中心にお話を聞くとともに、人材開発部部長の浅田恒夫氏に人材採用に対する同社の考えをうかがった。

積極姿勢が評価され 新潟と東京を仕事場に活躍

山田顕義氏の会社での肩書は「新潟支社システム課主任」。同社の新潟支社に中途採用されたのだが、パソコン・EWS関連のシステム開発を自ら希望して東京の情報システム事業部の仕事にも参加、いまは新潟と東京を往復しながら仕事をこなす。

「中途入社して3年目になります。以前は、都内にあるこの会社と同程度の規模のソフトウェアハウスに6年ほど勤めていました。SEをやっていましたが、メーカー色の強い会社でしたので、メーカーに入って制御系のシステムを担当するという仕事が中心でしたね。

私自身はパソコンやEWSの可能性に対して興味をもって見ていましたし、パッケージソフトなど自分が企画から関わって作りあげていくソフトづくりをしてみたいとも思っていました。それで別の会社に移ることを考えはじめたわけです。それと私は新潟出身なので、できれば地元で仕事のできる会社に、とも考えていました。

転職した当初は、新潟支店でオフコンのシステム関係の仕事をおもにしていましたが、パソコンやEWSのソフト開発の仕事がしたいという希望を出しまして、東京の事業部のほうでパッケージや企画商品的なものを開発している部署がありましたので、現在は新潟から月に何回か東京へ来て仕事をしています。」

会社全体から見ると、汎用機カラーが強い同社だが、システムインテグレーション業務やネットワーク関連への指向を高めている同社では、山田氏の積極性を高く評価したようだ。こうして新潟・東京を股にかけての仕事が始まった。

PC・マック・EWSまで 多彩に展開するソフト開発

現在山田氏は、情報システム事業部でさまざまな業種・企業向けのシステム開発にあたっている。

「現在企画している仕事としては、まずシルバー産業向けのものがひとつ。これはパソコン数台をNetWareを使用してLANで結んだグループウェアを作ろうというものです。現在シルバー産業は非常に脚光を浴びていますが、一方で社会的な責任も重い分野ですから、厚生省の指導も強化されつつあり、最近は経営的にきちんと数字を出さないと商売ができなくなってきたんですね。そこで、その辺りの市場性を狙って、コンサルティング会社とタイアップしてシステムを作ろうということです。パッケージというところまではいっていませんが、10本、20本という数で売っていこうと考えています。

それから、アップルのマッキントッシュ用のシステムにも取り組んでいます。いま進めているのは航空会社の経理や貨物の運送実績などを管理するシステムなどですね。

また、当社ではソニーのNEWSを使ったUNIXのシステムを以前から手掛けてきたんですが、私自身もNEWSには以前から興味を持っていましたので、航空会社向けのNEWSのソフトも開発しているんですよ。これは当社でデータ処理をしている毎月何千ページにもおよぶ航空会社の帳票類をもっと見やすいかたちに加工しようということで、データをCD−ROMに落とし、これをNEWS上で加工して出力するというソフトをいま作っているところです。

NEWSからPC98そしてマッキントッシュまでいろいろでしょ。いまはそれぞれのソフトの用途やマシンの特長を生かしたソフトづくりに挑戦しながら、積極的な姿勢で取り組むよう心がけています。」

好奇心を失ったら この仕事はつとまらない

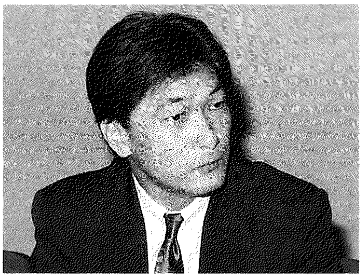
新しい分野へのチャレンジ精神旺盛な山田氏だが、プログラマ・SEの仕事に必要なものとはなんなのだろうか。

「まず、好奇心だと思います。とくに中途入社の場合は、時期的にも社内研修のプログラムに乗りにくいわけですから、自分で好奇心をもって、自分でものを覚えていく姿勢が大事でしょう。

それに、好奇心が大切というのはある程度キャリアのある人にも常に必要とされることじゃないでしょうか。技術的にどんどん進んでいる時代ですし、新しいものを積極的に取り込んでいこうという姿勢がないと。

もうひとつは、技術だけにとらわれない広い意味での判断力でしょうか。たとえばCOBOL系で入ってくれば、依然としてまだ大きい仕事が中心ですし、メーカーの提供した環境の中で標準化された工程に従って作っていくという作業なんですが、パソコンからワークステーション系の仕事になると事情は違ってきます。単位が小さいですから、プログラマーレベルは同様でも、SEとしてやっていくためには、たとえばお客様のところに出向いて、機器構成や開発ルーツの選定などからはじめて、マニュアル・パンフレットの原稿までタッチするということでないとつとまりません。ですから、あまり技術指向だけではいけないと思います。お客様にとって何が大事か、このくらいの商品性があれば売れるだろうといった、判断力ですね。そういうものが大事だと思いますね。」

山田氏の担当するプロジェクトでは、現在3人から5人ぐらいのチームを組み、ひとつのパッケージを1年間ぐらいの期間をかけて作っている。全社的にみればシェアはまだ小さいが、将来的には楽しみなことがたくさんあるという。



新潟支店システム課主任　山田顕義氏

「当社では、私の参加している情報システム事業部だけではなく、科学技術系や土木系の事業部でもパッケージソフトはやってきましたので、全社的に見ればある程度のパッケージソフト開発の実績はあると思います。

ただ、不特定のユーザー対象にソフト開発をするというのは、いままで当社ではあまりなかったので、現在はそういう仕事の要求があると私の担当になっています。チームはまだ少ないですが、パソコンやワークステーションレベルでの仕事がありますと、私のところに相談が持ち込まれますし、そういう点はうれしいところですね。

また、ダウンサイジングというテーマがいま非常に重要になってきていますから、最近はこれだけ小さい機械でここまでできる、といったことが他の部門への刺激になっていると思います。

今後は、まずふつうの受注ソフトの倍くらいの時間をかけて開発し、10〜20本程度出せる商品からはじめて、将来的にヒット商品を出せるようになりたいですね。」

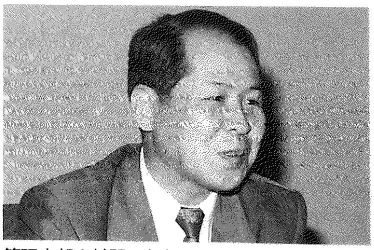
チャレンジする心を 尊重し伸ばす社風

最後に、人材開発部部長の浅田恒夫氏に同社の人材採用に対する考え方を語っていただいた。

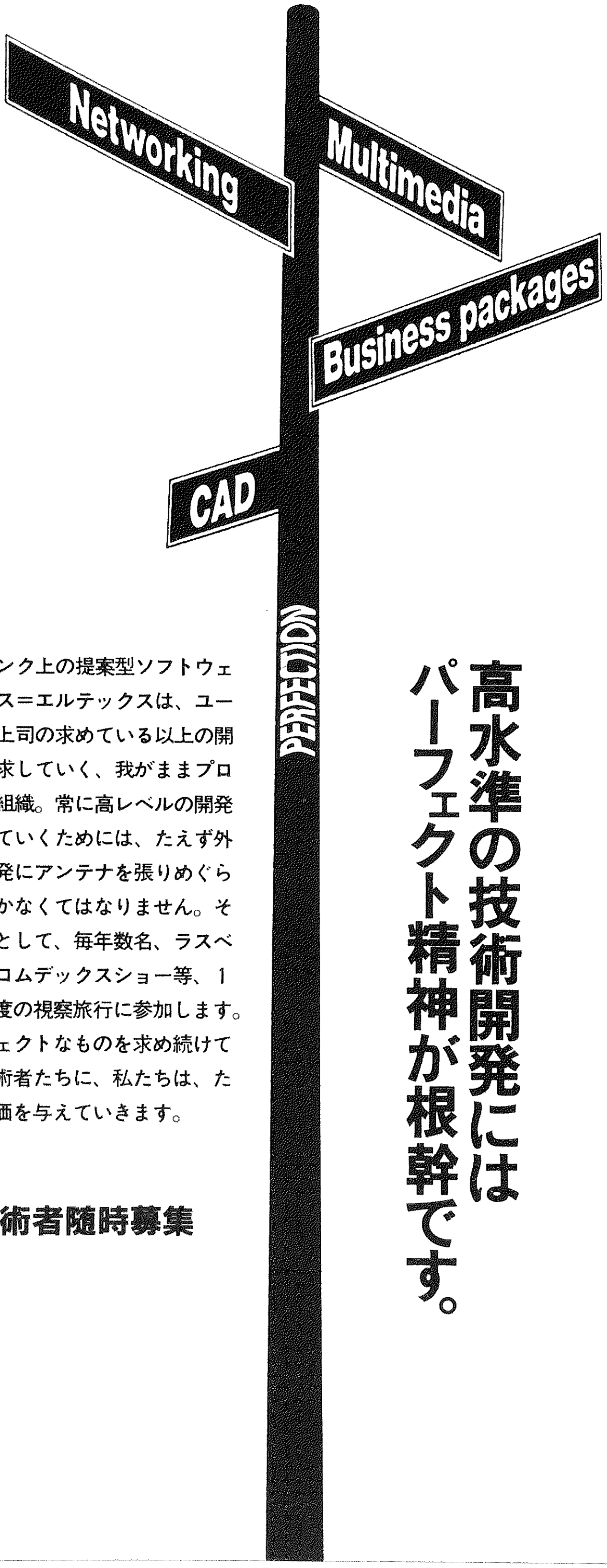
「当社ではここ3年で、毎年200人以上を採用しています。当社の従業員数は1500名余りですから、この数字がいかに大きいかがお分かりになるでしょう。当社はこれまで受託業務やシステム開発で安定した実績を築いてきましたが、さらに今後はシステムインテグレーション事業の高度化、EWSやネットワークの活用分野という、より大きなステージでの活躍をめざしているわけです。

この新しい時代を切り開くのは、もちろん若い人たちです。私が印象深く記憶している言葉があるんですが、それは前社長が『失敗の数が多いということは、それだけ何かを成功したということだ。失敗を恐れずにチャレンジしてほしい』と発言されたことです。〝失敗の数があなたのアクションの証しだ〟というんですね。単に技術力だけで評価するのではなく、当社にはそれぞれの人の〝心〟を大切にする社風があると思うんですよ。

ですから、若い人たちが意欲をもってどんどんチャレンジしてくれれば、われわれはそれに応えます。これが、当社の人材に対する考え方の根本だと思います。」



管理本部人材開発部部長　浅田恒夫氏



高水準の技術開発には パーフェクト精神が根幹です。

ワンランク上の提案型ソフトウェアハウス＝エルテックスは、ユーザーや上司の求めている以上の開発を追求していく、我がままプロ集団の組織。常に高レベルの開発を行っていくためには、たえず外部の開発にアンテナを張りめぐらしておかなくてはなりません。その一環として、毎年数名、ラスベガスのコムデックスショー等、1週間程度の視察旅行に参加します。パーフェクトなものを求め続けていく技術者たちに、私たちは、たえず評価を与えていきます。

技術者随時募集

会社概要

ELTEX

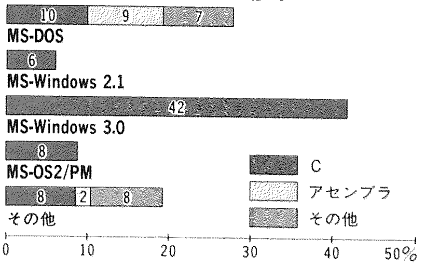
設　立	昭和60年12月14日
資本金	1,000万円
事業内容	パソコンを中心としたシステムの提案・開発・販売　①CD-ROM・LD など新しいメディアに対応したシステムの開発 ②ネットワークの端末システムの開発 ③特定業種向けCADの開発・販売 ④基本ソフトの開発
特　徴	大手クライアント向けのシステム開発が多く、ネットワーク関連システムやマルチメディアなど幅広く対応している。また、ＣＡＤを中心とした自社プロダクツにも積極的で、その比率も増加している。基本ソフトの分野では、ハードウェア専門のベンチャー企業とジョイントし、ＯＥＭ製品のＢＩＯＳ・ユーティリティなどを企画から手掛けている。
取引先	三菱電機、三菱自動車、日本航空　他
関連会社	株式会社ソーテック、学校法人福井情報ビジネス総合学院

募集要項

ELTEX

職　種	①システムエンジニア　②プログラマー
資　格	専門卒以上　20〜36歳の方　※Ｃ言語・アセンブラ言語経験者大歓迎　※未経験者でも好奇心・向学心の旺盛な方は歓迎します。
給　与	17万円（専門卒初任給）〜46万円（36歳課長、諸手当含む） 24歳例：23万5000円（情報処理１種手当・住宅手当含む） 30歳例：31万6000円（中級SE、情報処理１種・住宅手当含む）
手　当	時間外手当、役職手当（5000円〜６万円）、住宅手当（１万5000円）、資格手当（２種：5000〜特種：２万円）、通勤手当
賞　与	賞与年２回（平成１年度実績年間５ヵ月）、決算賞与年１回（平成１年度実績１ヵ月）
勤務地	本社（横浜駅近く） ◇派遣・出向はありません。
勤務時間	フレックスタイム制（コアタイム10:00〜16:00）試用期間中は標準時間9:00〜17:45
休日休暇	完全週休２日制（土・日）、祝日、年３回連休（GW・夏期・年末年始）、有給・特別・休暇
福利厚生	各種社会保険完備、退職金制度、財形貯蓄、社員旅行、海外研修、定期健康診断、社員寮、スポーツクラブ法人会員
応募方法	電話連絡の上、履歴書(写真貼付)、職務経歴書を持参し来社ください。郵送も受け付けます。
連絡先	〒220 横浜市西区南幸二丁目20番５号　東伸24ビル５階　℡(045)322-9667　総務課

最近のOS・言語別開発動向



採用HOT・LINE
TEL.(045)322-9667

株式会社エルテックス

12
DEC.1991

米国「The C Users Journal」「TECH Specialist」提携誌

月刊 Cジャーナル

C言語を主体とするコンピュータプログラマのための月刊情報誌

C JOURNAL

定価980円(本体951円)

12月号 11月8日発売

●連載●

★STANDARD C
★Dr.C's Pointers
★Questions & Answers
★Programming Style 第5回
★BART EXPRESS 第16回
★MINIX通信 第11回
★たのむぜ/DOS/V/ その⑤
★Pascal言語系の制作 第3回

特別記事 Turbo486のすすめ 兵藤嘉彦 MINIX386 on IBM-PC 木村孝道
スクリーンエディタFCOOE制作記 関谷 真

特集
アセンブラ&ディスアセンブラ
▶作者が語るSourcerとZZ386

付録 DISK
■FCO DEサンプル版
スクリーンエディタ
■CUG PDS #335
Frankenstein Cross Assemblers
■CUG PDS #348
8048 Disassembler/z80 Assembler
■掲載記事の全コード収録

1月号 12月8日発売 特集 移植記、IBM-PCから98へ

株式会社ビレッジセンター 出版局
〒101 東京都千代田区三崎町3－1－16 神田アメレックスビル2F
●営業部:03-3221-1825●編集部:03-3239-1950

C JOURNALは毎月8日全国の書店でお求めになれます。

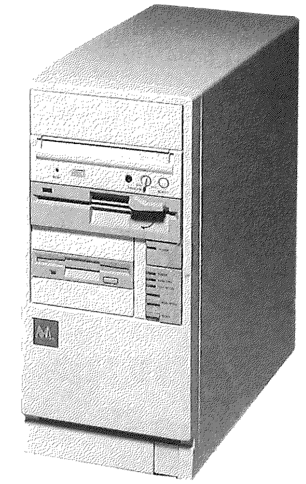
OTC 80486 MINI TOWER

¥556,000(税別、送料込み)

納期:3週間以内
保証:1年(国内保証)
CPU:80486 33MHz
CASH:256KB
RAM:4MB (1MB SIMM 70ns)
 UP TO 32MB ON BOARD
BIOS:AWORD
CHIP SET:INTEL
I/F:Serial*2/Parallel*1/SCSI*1
VIDEO:512KB SUPER VGA
(1024*768 16COLOR)
HDD:FUJITU 185MB SCSI
FDD:3.5" 1.44MB*1
KEYBOARD:101 TYPE

OPTION

4MB -> 8MB RAM	¥50,000
4MB -> 16MB RAM	¥120,000
4MB -> 32MB RAM	¥320,000
FUJITU 357MB 5" SCSI HDD	¥146,000
FUJITU 680MB 5" SCSI HDD	¥316,000
FUJITU 1.2GB 5" SCSI HDD	¥456,000
5" 1.2MB FDD	¥15,000
ATI VGA WONDER XL SUPER VGA CARD with MOUSE	¥55,000
HITACHI 14MVX 14" CRT (1024*768, 800*600 Non Interlaced)	¥85,000
IBM DOS 5.0	¥32,000
IBM DOS J4.0/V	¥18,400
MS MSDOS 5.0	¥17,500



お申し込みは、住所、氏名、電話番号等を明記の上、FAXまたは封書にて承ります。折り返し、振り込み先口座、納期等を御連絡いたします。官費、校費、カードによる支払い、分割払い等はお受けつけしておりません。悪しからずご了承ください。

広告有効期限 '91/11月末日

米国The C Users Group日本支部
VILLAGE CENTER INC.
■本部:〒101 東京都千代田区三崎町3-1-16神田アメレックスビル2F
TEL 03-3239-2187 FAX 03-3221-1768
■Shop:〒101 東京都千代田区神田神保町1-35-15 たるまビル1F
TEL 03-3219-6113

スタッフ募集中!!▶明るくヤル気のある方大歓迎です。詳しくは電話3239-2187 担当:中村までお問い合わせください。

QuickBASIC Ver.4.5ステートメント・関数一覧(PC-9800シリーズ・各社AX対応)

●ABS - 数式の絶対値を返す数値演算関数
構文：ABS(numeric-expression)

●ASC - 引数として指定した文字列の，最初の文字のASCIIコードまたは，漢字連続コードを返す文字列処理関数
構文：ASC(stringexpression)

●ATN - 引数のアークタンジェント（タンジェントの逆関数）を返す数値演算関数
構文：ATN(numeric-expression)

●BEEP - スピーカーを鳴らすデバイスI/Oステートメント
構文：BEEP

●BLOAD - BSAVEステートメントで作ったメモリイメージファイルを，入力ファイルまたはデバイスから，メモリに読み込むファイルI/Oステートメント
構文：BLOAD filespec [, offset]

●BSAVE - あるメモリ領域の内容を，ファイルまたはデバイスに転送するファイルI/Oステートメント
構文：BSAVE filespec, offset, length

●CALL - BASICのSUBプロシージャに制御を渡すステートメント
構文：CALL name [(argumentlist)]
name [(argumentlist)]

●CALL, CALLS - 他の言語で書かれたプロシージャに制御を渡すステートメント
構文：CALL name [(call-argumentlist)]
name [call-argumentlist]
CALLS name [(calls-argumentlist)]

●CALL ABSOLUTE - 機械語のプロシージャに制御を渡すステートメント
構文：CALL ABSOLUTE([argumentlist,]integervariable)

●CALL INTERRUPT - QuickBASICのプログラムからDOSのシステムコールやBIOSコールなどを実行するサブプログラム
構文：CALL INTERRUPT(interruptnum, inregs, outregs)
CALL INTERRUPTX(interruptnum, inregs, outregs)

●CDBL - 引数に指定した数式を，倍精度の数値に変換する関数
構文：CDBL(numeric-expression)

●CDBL$ - 文字列中の1バイト系文字を，2バイト系文字に変換
構文：CDBL$(stringexpression)

●CHAIN - 他のプログラムに制御を渡すプログラム制御ステートメント
構文：CHAIN filespec

●CHDIR - 指定したドライブのデフォルトディレクトリを変更するステートメント
構文：CHDIR pathspec

●CHR$ - 引数として指定したASCIIコード，漢字連続コードに対応する文字を返す文字列処理関数
構文：CHR$(code)

●CINT - 引数に指定した数式の小数部を丸めて，整数に変換する関数
構文：CINT(numeric-expression)

●CIRCLE - 指定した中心と半径を持つ，楕円または円を描くグラフィックスステートメント
構文：CIRCLE [STEP] (x,y), radius[,[color][,[start][,[end][,aspect]]]]

●CLEAR - すべての変数を初期化し，ファイルをクローズするメモリ管理ステートメント，スタックサイズの設定も可能
構文：CLEAR [, , stack]

●CLNG - 引数に指定した数式の小数部を丸めて，4バイトの長整数に変換する関数
構文：CLNG(numeric-expression)

●CLOSE - ファイルやデバイスへの入出力を終了するファイルI/Oステートメント
構文：CLOSE [[#] filenumber [, [#] filenumber]...]

●CLS - ディスプレイ画面をクリアするデバイスI/Oステートメント
構文：CLS [{0 | 1 | 2}]

●COLOR - 表示色を指定するグラフィックスステートメント
構文：COLOR [foreground][,background]

●COM(n) - 指定したポートでの通信イベントのトラッピングを開始／中止，中断するイベントトラッピングステートメント
構文：COM (n) ON
COM (n) OFF
COM (n) STOP

●COMMAND$ - DOSからプログラムを呼び出したときの，コマンドライン上で指定された引数リストを返す関数
構文：COMMAND$

●COMMON - モジュールの間やチェインしたプログラムの間で，グローバル変数の共有を宣言するステートメント
構文：COMMON [SHARED][/blockname/] variablelist

●CONST - 数値や文字列値の代わりに使う，記号定数を宣言するステートメント
構文：CONST constantname = expression [, constantname = expression]...

●COS - 引数に指定した数値のコサインを返す数値演算関数
構文：COS(numeric-expression)

●CSNG - 引数として指定した数式を，単精度の数値に変換する関数
構文：CSNG(numeric-expression)

●CSNG$ - 文字列中の2バイト文字を1バイト文字に変換する関数
構文：CSNG$(stringexpression)

●CSRLIN - 現在，カーソルが位置する行を返すデバイスI/O関数
構文：CSRLIN

●CVSMBF , CVDMBF - Microsoftバイナリ形式の数値を含む文字列を，IEEE形式の数値に変換する関数
構文：CVSMBF (4-byte-string)
CVDMBF (8-byte-string)

●CVI, CVS, CVL, CVD - 数値内部表現の文字列を，実際の数値に変換する関数
構文：CVI(2-byte-string)
CVS(4-byte-string)
CVL(4-byte-string)
CVD(8-byte-string)

●DATA - READ ステートメントで読み込む数値定数や，文字列定数を定義するステートメント
構文：DATA constant [, constant2]...

●DATE$ - 現在の日付を含む文字列を返す関数
構文：DATE$

●DATE$ - 現在の日付を設定するステートメント
構文：DATE$ = stringexpression

●DECLARE - QuickBASICのプロシージャへの参照を宣言し，引数のデータ型をチェックするステートメント
構文：DECLARE {FUNCTION | SUB} name [([parameterlist])]

●DECLARE - 他の言語で書いた外部プロシージャを呼び出す手順を宣言するステートメント
構文：DECLARE FUNCTION name [CDEL] [ALIAS "aliasname"][([parameterlist])]
DECLARE SUB name [CDEL] [ALIAS "aliasname"][([parameterlist])]

●DEF FN - 関数を定義し，名前を付けるステートメント
構文1
DEF FN name [(parameterlist)] = expression
構文2
DEF FN name [(parameterlist)]
[statements]
FNname = expression
[statements]
END DEF

●DEF SEG - 後に続くBLOAD, BSAVE, CALL ABSOLUTE, POKEステートメントやPEEK関数で参照するセグメントアドレスを設定するステートメント
構文：DEF SEG [=address]

●DEFtype - 変数，DEF FN関数，FUNCTIONプロシージャのデフォルトのデータ型を設定するステートメント
構文：DEFINT letterrange [,letterrange]...
DEFSNG letterrange [,letterrange]...
DEFDBL letterrange [,letterrange]...
DEFLNG letterrange [,letterrange]...
DEFSTR letterrange [,letterrange]...

●DIM - 複数の変数名を宣言し，それらにメモリ領域を割り当てる宣言ステートメント
構文：DIM [SHARED] variable [(subscripts)] [AS type][, variable [(subscripts)] [AS type]...

●DO...LOOP - 指定した条件が真である間，または指定した条件が真になるまで，一連のステートメントの実行を繰り返すフロー制御ステートメント
構文1
DO [{WHILE | UNTIL} booleanexpression]
[statementblock]
LOOP
構文2
DO
[statementblock]
LOOP [{WHILE | UNTIL} booleanexpression]

●DRAW - 引数として指定した文字列式を解釈し，図形を描くグラフィックスステートメント
構文：DRAW stringexpression

●END - プログラム，プロシージャ，ブロックを終了する宣言
構文：END [{DEF | FUNCTION | IF | SELECT | SUB | TYPE}]

●ENVIRON$ - DOSの環境文字列テーブルから，指定した環境文字列を検索する関数
構文：ENVIRON$ (environmentstring)
ENVIRON$ (n)

●ENVIRON - DOSの環境文字列テーブル内のパラメータを変更するステートメント
構文：ENVIRON stringexpression

●EOF - ファイルの終りを調べる，ファイルI/Oステートメント
構文：EOF(filenumber)

●ERASE - STATIC 配列の要素を再初期化したり，DYNAMIC配列に割り当てたメモリを解放するメモリ管理ステートメント
構文：ERASE arrayname [, arrayname...]

●ERDEV , ERDEV$ - エラーが発生した後で，デバイスの状態を知らせるエラートラッピング関数
構文：ERDEV
ERDEV$

●ERL , ERR - エラーが発生した後で，プログラムの状態を知らせるエラートラッピング関数
構文：ERR
ERL

●ERROR - エラーを疑似的に発生させたり，ユーザーエラーコードを定義するエラートラッピングステートメント
構文：ERROR integerexpression

●EXIT - DEF FN関数，DO...LOOP，FOR...NEXTループ，FUNCTION，SUBプロシージャを抜けるフロー制御ステートメント
構文：EXIT {DEF | DO | FOR | FUNCTION | SUB}

●EXP - 指数関数（eを底とする数式のべき乗）を計算する数値演算関数
構文：EXP(numeric-expression)

●FIELD - ランダムアクセスファイルのバッファ内に，変数のための領域を割り当てるファイルI/Oステートメント
構文：FIELD [#]filenumber, fieldwidth AS stringvariable[,fieldwidth AS stringvariable] ...

●FILEATTR - オープンしている任意のファイルの情報を返すファイルI/O関数
構文：FILEATTR (filenumber, attribute)

●FILES - 指定したディスク上にあるファイルの一覧を表示するステートメント
構文：FILES [filespec]

●FIX - 引数として指定した数式の小数部を切り捨て，整数部だけを返す数値演算関数
構文：FIX(numeric-expression)

●FOR...NEXT - 指定した回数だけ，一連のステートメントを実行するフロー制御ステートメント
構文：FOR counter = start TO end [STEP increment]
[statements]
NEXT [counter [, counter...]]

●FRE - 使用可能なメモリ容量を返すメモリ管理関数
構文：FRE(stringexpression)
FRE(numeric-expression)

●FREEFILE - 次に使用可能なファイル番号を返すファイルI/O関数
構文：FREEFILE

●FUNCTION...END FUNCTION - FUNCTIONプロシージャを構成する，名前，パラメータ，コードを宣言するステートメント
構文：FUNCTION name [(parameterlist)] [STATIC]
[statements]
name = expression
[statements]
END FUNCTION

●GET - ディスクファイルの内容を，ランダムアクセスファイルのバッファや，変数に読み込むファイルI/Oステートメント
構文：GET [#] filenumber [, [recordnumber] [,variable]]

●GET - 画面上のグラフィックスイメージを，配列内に格納するステートメント
構文：GET [STEP] (x1,y1) - [STEP] (x2,y2), arrayname [(indices)]

●GOSUB...RETURN - サブルーチンへの分岐と，サブルーチンからの復帰を行うフロー制御ステートメント
構文：GOSUB {linelabel1 | linenumber1}
[statements]
RETURN [linelabel2 | linenumber2]

●GOTO - 指定した行へ無条件に分岐するフロー制御ステートメント
構文：GOTO {linelabel | linenumber}

●HEX$ - 引数に指定した10進数を，16進で表現した文字列に変換する文字列処理関数
構文：HEX$(numeric-expression)

●IF...THEN...ELSE - 任意の式を評価した値に基づいて，条件付きの実行や分岐を行うフロー制御ステートメント
構文1（ブロック形式）
IF booleanexpression1 THEN
[statementblock-1]
[ELSEIF booleanexpression2 THEN
[statementblock-2]]
...
[ELSE
[statementblock-n]]
END IF
構文2（1行の形式）
IF booleanexpression THEN thenpart [ELSE elsepart]

●INKEY$ - キーボードから文字を読み込む，デバイスI/Oステートメント
構文：INKEY$

●INP - I/Oポートから読み込んだ1バイトのデータを返すデバイスI/O関数
構文：INP(port)

●INPUT$ - 指定したファイルから，読み出した文字列を返すファイルI/O関数
構文：INPUT$(n [,[#] filenumber])

●INPUT¥ - 指定したファイルから，指定した長さの文字列を読み込む
構文：INPUT¥(n[, [#] filenumber])

●INPUT - プログラムの実行中，キーボードから入力されたデータを読み込み，それを変数リストの中に格納するデバイスI/Oステートメント
構文：INPUT [;] ["promptstring" {; | ,}] variablelist

●INPUT# - シーケンシャルデバイスやシーケンシャルファイルから，データ項目（フィールド）を読み込み，その値を変数に代入するステートメント
構文：INPUT #filenumber, variablelist

●INSTR - ある文字列の中から別の文字列を検索し，最初に見つかった位置を返す文字列処理関数
構文：INSTR([start,]stringexpression1, stringexpression2)

●INT - 引数に指定した数式以下で，最大の整数を返す数値演算関数
構文：INT(numeric-expression)

●IOCTL$ - デバイスドライバから制御データ文字列を受け取るファイルI/O関数
構文：IOCTL$([#] filenumber)

●IOCTL - デバイスドライバへ制御データ文字列を送るファイルI/Oステートメント
構文：IOCTL [#] filenumber, string

●JIS$ - 文字列の先頭1文字のJISコードを文字列として返す関数
構文：JIS$(stringexpression)

●KEXT$ - 文字列から1バイト系，あるいは2バイト系文字だけを抜き出す関数
構文：KEXT$(stringexpression, func)

●KEY - ファンクションキーに，"ソフトキー"文字列を割り当てるデバイスI/Oステートメント
構文：KEY LIST　　ソフトキー文字列の値を画面に表示
KEY {ON | OFF}　　画面最下行のソフトキーの表示／非表示を設定
KEY n,stringexpression　　ソフトキーに文字列を割り当てる

●KILL - ディスクファイルを削除するステートメント
構文：KILL filespec

●KINSTR - 文字列の中から指定した文字列を探し，その位置を返す
構文：KINSTR ([start], stringexpression1, stringexpression2)

●KLEN - 文字列の文字数を返す
構文：KLEN(stringexpression)

●KMID$ - 文字列の一部を，別の文字列で置き換えるステートメント
構文：KMID$(stringvariable, start [, count])=stringexpression

●KMID$: 文字列の中から，指定した文字数分の文字列を返す関数
構文：KMID$(stringvariable, start [, count])

●KPOS - 文字列中の指定した文字位置までの，バイト数を返す関数
構文：KPOS(stringvariable, characternumber)

●KTN$ - 文字列の先頭の1文字を句点コードまたはアスキー文字コードに変換する関数
構文：KTN$(stringexpression)

●LBOUND - 配列の指定した次元で使うことができる，添字の最小値を返すメモリ管理関数
構文：LBOUND(array [, dimension])

●LCASE$ - 指定した文字列中の，すべての英字を小文字に変換する文字列処理関数
構文：LCASE$(stringexpression)

●LEFT$ - 指定した文字列の左端から，nバイトの文字列を取り出す文字列処理関数
構文：LEFT$(stringexpression, n)

●LEN - 指定した文字列のバイト数，または，変数に必要なバイト数を返す文字列処理関数
構文：LEN(stringexpression)
LEN(variable)

●LET - 数式を計算して，その値を変数に代入するステートメント
構文：[LET] variable=expression

●LINE - 画面上に直線や四角形を描くグラフィックスステートメント
構文：LINE [[STEP] (x1,y1)] - [STEP] (x2,y2) [, [color][, [B [F]] [, style]]]

●LINE INPUT - キーボードから入力された行全体を読み込んで，1つの文字列変数に格納するステートメント
構文：LINE INPUT [;] ["promptstring";] stringvariable

●LINE INPUT# - シーケンシャルファイルから区切り文字を含まない行全体を読み込んで，1つの文字列変数に格納するステートメント
構文：LINE INPUT #filenumber, stringvariable

●LOC - ファイル内の，次に読み書きする位置を返すファイルI/O関数
構文：LOC(filenumber)

●LOCATE - 指定した位置にカーソルを移動するデバイスI/Oステートメント
構文：LOCATE [row] [, [column] [, [cursor] [, [start, stop]]]]

●LOCK...UNLOCK - オープンしているファイルの一部または全部に対する，他のプロセスからのアクセスを制御するファイルI/Oステートメント
構文：LOCK [#] filenumber [, {record | [start] TO end}]
[statements]
UNLOCK [#] filenumber [, {record | [start] TO end}]

●LOF - 指定したファイルの長さをバイト単位で返すファイルI/Oステートメント
構文：LOF(filenumber)

●LOG - 引数として指定した数式の自然対数を返す数値演算関数
構文：LOG(numeric-expression)

●LPOS - プリンタバッファ内のプリンタヘッドの現在位置を返すデバイスI/O関数
構文：LPOS(n)

●LPRINT, LPRINT USING - プリンタLPT1:にデータを印刷するファイルI/Oステートメント
構文1
LPRINT [expressionlist] [{; | ,}]
構文2
LPRINT USING formatstring; expressionlist [{; | ,}]

●LSET - PUTステートメントを使ってデータを書き込むとき，データをランダムアクセスファイルバッファに格納するためのファイルI/Oステートメント，文字列変数の値を左詰めにしたり，レコード変数を別のレコード変数にコピーする
構文：LSET stringvariable=stringexpression

●LTRIM$ - 指定した文字列から先頭のスペースを削除した文字列を返す関数
構文：LTRIM$(stringexpression)

●MID$ - 文字列変数の一部を別の文字列に置き換える文字列処理ステートメント
構文：MID$(stringvariable, start [, length]) = stringexpression

●MID$ - 指定した文字列の一部を取り出す関数
構文：MID$(stringexpression, start [, length])

●MKDIR - DOSを利用して，新しいディレクトリを作るステートメント
構文：MKDIR pathname

●MKSMBF$, MKDMBF$ - IEEE形式の数値を，Microsoftバイナリ形式の文字列に変換する関数
構文：MKSMBF$(single-precision-expression)
MKDMBF$(double-precision-expression)

●MKD$, MKI$, MKL$, MKS$ - 数値を文字列値に変換する関数
構文：MKI$(integerexpression)
MKS$(single-precision-expression)
MKL$(long-integer-expression)
MKD$(double-precision-expression)

●NAME - DOSを利用して，ディスクファイルやディレクトリの名前を変えるステートメント
構文：NAME oldfilename AS newfilename

●OCT$ - 引数として指定した数値を，8進数で表現した文字列に変換する文字列処理関数
構文：OCT$(numeric-expression)

●ON ERROR - エラー処理を可能にし，エラー処理ルーチンの最初の行を指定するステートメント
構文：ON ERROR GOTO line

●ON event - イベントトラッピングルーチンの最初の行を指定するステートメント
構文：ON event GOSUB {linenumber | linelabel}

●ON UEVENT GOSUB - ユーザー定義のイベントをトラッピングするサブルーチンを定義するステートメント
構文：ON UEVENT GOSUB {linenumber | linelabel}

●ON...GOSUB, ON...GOTO - 式の値に応じて，指定した行番号／行ラベルリストのひとつに分岐するステートメント
構文：ON expression GOSUB {line-number-list | line-label-list}
ON expression GOTO {line-number-list | line-label-list}

●OPEN - ファイルやデバイスへの入出力を可能にするステートメント
構文：OPEN file [FOR mode1] [ACCESS access] [lock] AS [#] filenum [LEN = reclen]
OPEN mode2, [#] filenum, file [, reclen]

●OPEN COM - 入出力を行うため通信チャネルをオープンして初期化するステートメント
構文：OPEN "COMn: optlist1 optlist2" [FOR mode] AS [#]filenum [LEN = reclen]

●OPTION BASE - 配列の添字の最小値を宣言するステートメント
構文：OPTION BASE n

●OUT - I/Oポートに1バイトのデータを送るデバイスI/Oステートメント
構文：OUT port, data

●PAINT - 指定した色や模様を使って，領域を塗りつぶすグラフィックスステートメント
構文：PAINT [STEP] (x,y) [, [paint] [, [bordercolor] [, background]]]

●PALETTE, PALETTE USING - パレットの色を変えるグラフィックスステートメント
構文：PALETTE [attribute, color]
PALETTE USING array-name [(array-index)]

●PCOPY - あるスクリーンページを，別のページにコピーするグラフィックスステートメント
構文：PCOPY sourcepage, destinationpage

●PEEK - メモリ上の指定した位置に格納されているバイトデータを返すメモリ管理関数
構文：PEEK(address)

●PLAY - BGMキューに含まれる音符の数を返す関数
構文：PLAY (n)

●PLAY - 音楽を演奏するデバイスI/Oステートメント
構文：PLAY commandstring

●PLAY ON, OFF, STOP - BGMのイベントトラッピングを可能／中止／中断するステートメント
構文：PLAY ON
PLAY OFF
PLAY STOP

●PMAP - 論理座標を物理座標に，物理座標を論理座標に変換するグラフィックス関数
構文：PMAP (expression, function)

●POINT - 画面上のピクセルの色番号の読み取りや，ピクセルの座標を返すグラフィックス関数
構文：POINT (x,y)
POINT (number)

●POKE - メモリの指定した位置にバイトデータを書き込むメモリ管理関数
構文：POKE address, byte

●POS - 現在，カーソルの桁位置を返すデバイスI/O関数
構文：POS(0)

●PRESET - 画面上の指定する座標に，点を描くグラフィックスステートメント
構文：PRESET [STEP] (x,y) [, color]

●PRINT - 画面上にデータを出力するデバイスI/Oステートメント
構文：PRINT [expressionlist] [{, | ;}]

●PRINT #, PRINT # USING - データをシーケンシャルファイルに書き込むステートメント
構文：PRINT #filenumber, [USING stringexpression;] expressionlist [{, | ;}]

●PRINT USING - 指定した書式を使って文字列や数値を画面に表示するステートメント
構文：PRINT formatstring; expressionlist [{, | ;}]

●PSET - 画面上に点を描くグラフィックスステートメント
構文：PSET [STEP] (x,y) [, color]

●PUT - 変数や，ランダムアクセスファイルのバッファの内容を，ファイルに書き込むファイルI/Oステートメント
構文：PUT [#] filenumber [, [recordnumber] [, variable]]

●PUT - GETステートメントで保存したグラフィックスイメージを画面上に表示するステートメント
構文：PUT [STEP] (x,y), arrayname [(indices)] [, actionverb]

●RANDOMIZE - 乱数ジェネレータを初期化（乱数系列を再設定）する数値演算ステートメント
構文：RANDOMIZE [expression]

●READ - DATAステートメントから値を読み込み，変数に代入するI/Oステートメント
構文：READ variablelist

●REDIM - $DYNAMIC 配列に割り当てた領域を変更する宣言
構文：REDIM [SHARED] variable(subscripts) [AS type] [, variable (subscripts) [AS type]]...

●REM - プログラムに注釈を挿入するステートメント
構文：REM remark
' remark

●RESET - すべてのディスクファイルをクローズするファイルI/Oステートメント
構文：RESET

●RESTORE - 指定した行にある DATAステートメントから，データを再び読み込むためのI/Oステートメント
構文：RESTORE [{linelabel | linenumber}]

●RESUME - エラートラッピングルーチンが呼び出された後で，プログラムの実行を継続するエラートラッピングステートメント
構文：RESUME [0]
RESUME NEXT
RESUME {linelabel | linenumber}

●RETURN - サブルーチンから制御を戻すフロー制御ステートメント
構文：RETURN [{linelabel | linenumber}]

●RIGHT$ - 指定した文字列の右端から，n個の文字を取り出す文字列処理関数
構文：RIGHT$(stringexpression, n)

●RMDIR - 指定したディレクトリを削除するステートメント
構文：RMDIR pathname

●RND - 0から1の範囲の，単精度の乱数を返す数値演算関数
構文：RND[(n)]

●RSET - PUTステートメントを使ってデータを書き込むとき，データをランダムアクセスファイルバッファに格納するためのファイルI/Oステートメント，このステートメントは，文字列変数の値を右詰めにする
構文：RSET stringvariable = stringexpression

●RTRIM$ - 指定した文字列の末尾の全角も含めたスペースを削除する文字列処理関数
構文：RTRIM$(stringexpression)

●RUN - 現在，メモリ上にあるプログラムをリスタートしたり，指定したプログラムを実行するフロー制御ステートメント
構文：RUN [{linenumber | commandline}]

●SADD - 指定した文字列式のアドレスを返すメモリ管理関数
構文：SADD (stringvariable)

●SCREEN - 指定した画面上の位置にある文字の，ASCIIコードや色を読み取る関数
構文：SCREEN(row, column [, colorflag])

●SCREEN - 画面の設定を行うグラフィックスステートメント
構文：SCREEN [mode] [, [paletteswitch]] [, [apage]] [, [vpage]]

●SEEK - ファイル内の，現在入出力を行っている位置を返す関数
構文：SEEK(filenumber)

●SEEK - ファイル内の，次に読み書きする位置を設定するファイルI/Oステートメント
構文：SEEK [#] filenumber, position

●SELECT CASE - 条件式の値に従って，複数のステートメントブロックのいずれかを実行させるフロー制御ステートメント
構文：SELECT CASE testexpression
[CASE expressionlist1]
[statementblock-1]
[CASE expressionlist2
[statementblock-2]] ...
[CASE ELSE
[statementblock-n]]
END SELECT

●SETMEM - farヒープで使うメモリ容量を変更するメモリ管理関数，farヒープとは，farオブジェクトと内部テーブルを格納する領域のこと
構文：SETMEM(numeric-expression)

●SGN - 指定した数式の符号を返す数値演算関数
構文：SGN(numeric-expression)

●SHARED - モジュールレベルで宣言した変数をパラメータとして引き渡さずに，SUB，FUNCTIONプロシージャでアクセスするためのステートメント
構文：SHARED variable [AS type] [, variable [AS type]]...

●SHELL - 実行中のBASICのプログラムを抜けて，.COM, .EXE, .BATプログラムやDOSコマンドを実行するステートメント，実行を終了すると，SHELLステートメントの次の行に制御が戻る
構文：SHELL [commandstring]

●SIN - ラジアンを単位とする角度のサインを返す数値演算関数
構文：SIN(numeric-expression)

●SLEEP - プログラムの実行を一時中断するフロー制御ステートメント
構文：SLEEP [seconds]

●SOUND - スピーカーから音を発生させるデバイスI/Oステートメント
構文：SOUND frequency, duration

●SPACE$ - n個のスペースからなる文字列を返す文字列処理関数
構文：SPACE$(n)

●SPC - PRINT, LPRINTステートメント中で，n個のスペースを表示するデバイスI/O関数
構文：SPC(n)

●SQR - 引数に指定した数式の平方根を返す数値演算関数
構文：SQR(numeric-expression)

●STATIC - 単純変数や単純配列を，あるプロシージャ（またはDEF FN関数）に対して，ローカルな変数として宣言するステートメント，ローカル変数は，次にプロシージャが呼び出されるまで，変数の値を保存する
構文：STATIC variablelist

●STOP - プログラムの実行を中止するフロー制御ステートメント
構文：STOP

●STR$ - 数式の値を文字列で表現した値を返す文字列処理関数
構文：STR$(numeric-expression)

●STRING ON, OFF, STOP - ジョイスティックのイベントトラッピングをオン／オフしたり，中断したりする．
構文：STRING(n) ON
STRING(n) OFF
STRING(n) STOP

●STRING$ - 指定した文字コード，または文字列の先頭文字から成る文字列を返す文字列処理関数
構文：STRING$(m, numeric-expression)
STRING$(m, stringexpression)

●SUB...END SUB - SUBプロシージャの始めと終りを示すステートメント
構文：SUB globalname [parameterlist] [STATIC]
[statements]
[EXIT SUB]
[statements]
END SUB

●SWAP - 2つの変数の値を交換する代入ステートメント
構文：SWAP variable1, variable2

●SYSTEM - オープンしているすべてのファイルをクローズし，オペレーティングシステムに制御を戻すステートメント
構文：SYSTEM

●TAB - PRINT, LPRINTステートメントの中で，データの表示位置を移動するデバイスI/O関数
構文：TAB(column)

●TAN - ラジアン単位で指定した角度のタンジェントを返す数値演算関数
構文：TAN(numeric-expression)

●TIME$ - オペレーティングシステムから現在の時刻を取得する関数
構文：TIME$

●TIME$ - DOSを利用して，現在の時刻を設定するステートメント
構文：TIME$ = stringexpression

●TIMER ON, OFF, STOP - タイマイベントのトラッピングを，開始，中断，中止するステートメント
構文：TIMER ON

●TIMER - 午前0時からそれまでに経過した秒数を返します．
構文：TIMER ON
TIMER OFF
TIMER STOP

●TRON , TROFF - プログラムステートメントの実行をトレースする，デバッグのためのステートメント
構文：TRON
TROFF

●TYPE - 1つ以上の要素を含むユーザー定義のデータ型を宣言するステートメント
構文：TYPE usertype
elementname AS typename
[elementname AS typename]
...
END TYPE

●UBOUND - 配列の指定した次元で使うことができる，添字の最大値を返すメモリ管理関数
構文：UBOUND(array [, dimension])

●UCASE$ - 指定した文字列中の全角文字も含めたすべての英字を大文字に変換する文字列処理関数
構文：UCASE$(stringexpression)

●UEVENT - ユーザー定義イベントのイベントトラッピングを開始，中断，中止するステートメント
構文：UEVENT ON
UEVENT OFF
UEVENT STOP

●UNLOCK - ファイルに設定されたロックを解除するファイルI/Oステートメント
構文：UNLOCK [#] filenumber [, {record | [start] TO end}]

●VAL - 文字列で表現された値を数値に変換する文字列処理関数
構文：VAL (stringexpression)

●VARPTR , VARSEG - 変数のアドレスを返すメモリ管理関数
構文：VARPTR(variablename)
VARSEG(variablename)

●VARPTR$ - DRAW, PLAYステートメントで使うため，変数のアドレスを文字列の形で返すメモリ管理関数
構文：VARPTR$(variablename)

●VIEW - グラフィックスの出力範囲を定義するグラフィックスステートメント
構文：VIEW [[SCREEN] (x1,y1)-(x2,y2) [, [color] [, border]]]

●VIEW PRINT - 画面上のテキストビューポートの境界を設定するステートメント
構文：VIEW PRINT [topline TO bottomline]

●WAIT - プログラムの実行を一時中断し，入力ポートの状態を調べるフロー制御ステートメント
構文：WAIT portnumber, and-expression [, xor-expression]

●WHILE...WEND - 指定した条件が真である間，ループ内の一連のステートメントを実行するフロー制御ステートメント
構文：WHILE condition
[statements]
WEND

●WIDTH - ファイルやデバイスに出力する行の幅を割り当てたり，画面に表示する桁数や行数を設定するファイルI/Oステートメント
構文：WIDTH [columns] [, lines]
WIDTH {#filenumber | device}, width
WIDTH LPRINT width

●WINDOW - ビューポートの論理座標を定義するグラフィックスステートメント
構文：WINDOW [[SCREEN] (x1,y1) - (x2,y2)]

●WRITE - 画面にデータを出力するステートメント
構文：WRITE [expressionlist]

●WRITE # - シーケンシャルファイルにデータを書き込むファイルI/Oステートメント
構文：WRITE #filenumber [,expressionlist]

i80386／486 32-bit専用DOSエクステンダー

DOS/4G

▲386ネイティブ・ソースレベル・デバッガ標準装

4ギガバイト・メモリ空間のアドバンテージ

32-bitパフォーマンスをDOS上で実現！

DOS/4Gは、Intel80386/486CPUが持つ32-bitパフォーマンスを、DOSプログラムからダイレクトに活用可能にする、80386/486CPU専用DOSエクステンダです。DOS/4Gにより、DOSプログラムで最大4ギガバイトまでのメモリが使用可能になります。また、32-bitレジスタを使用してセグメント切り替えの無いフラットなメモリ・アクセスが実現されるため、従来の16-bitプログラムに比較して約50%アクセス・スピードを向上できます。

DOS/4Gが組み込まれたアプリケーションは32-bit・ネイティブ・プログラムでありながら通常のDOS環境でそのまま動作します。しかも、WINDOWS等他のメモリ拡張手段に比べて、既存DOSアプリケーションからの改変作業は最小限で済ませることができます。32-bitCPUのメモリ・パフォーマンスをDOS/4Gでフルに活用することによって、従来のパソコン用アプリケーション・プログラムのパフォーマンス向上だけでなく、ワークステーション上の大規模プログラムのパソコン上への移行が可能になります。

■DPMI規格に完全準拠

● 80386プロテクトモードとDOSとのインターフェース標準規格DPMI(DOS Protected Mode Interface)に完全準拠しています。

● 従って、DOS/4Gで作成されたプログラムは、リアル・モードDOS環境下以外でもWINDOWS 3を始めとするDPMI規格準拠のホスト下で動作、または、共存が可能です。

■32-bitネイティブ・モード専用 高機能ソース・レベル・デバッガD/32を装備

● 32-bitアプリケーション EXE ファイルを、80386/486ネイティブ・プロテクトモード上でデバッグするための、ウインドウタイプ高機能デバッガです。

● DOS/4Gで作成された32-bitプログラムが、ソース・レベルでCodeViewと同様な感覚でデバッグできます。(右上画面写真参照)

■32-bitネイティブ・モード専用リンカーGLU/32を装備

● マイクロソフトLINKと互換性を保ったスクリプト・ファイルや、オプションが使用できます。

■仮想メモリ・マネージャVMM(別売)を用意

● DOS/4Gのオプション製品であるVMMは、パソコンに実装されている以上のメモリ空間を使用するアプリケーションの実行を可能にします。

● VMMは80386/486のページング機能を活用し、コードとデータを自動的にディスクへスワップさせます。

● VMMのAPIは、業界標準DPMI 1.0に含まれる仮想メモリAPIのスーパセットであり、上位互換性があります。

■業界標準のDOSエクステンダ

● 16-bitDOSエクステンダである「DOS/16M」は、Lotus、IBM、Ashton-Tate、DEC、AutoDesk、InformiXなど、300社をこえるソフトウェア開発会社の主要製品に用いられ、すでに数百万コピーのアプリケーションに組み込まれて出荷されています。

● DOS/4Gは、このDOS/16Mの上に構築されており、32-bitネイティブ・アプリケーション開発のためのアップグレード・パスとなります。

32ビット・16ビットDOSエクステンダ製品仕様

	新発売 DOS/4G	DOS/16M
対応機種	▶PC-9801 ▶PC/AT及び互換機 (但し、それぞれ80386以上搭載機)	▶PC-9801 ▶FMR60/70 ▶B16H×II/32GX ▶if800/RX,EX ▶PC/AT及び互換機(但し、それぞれ80286以上搭載機)
対応言語	▶WATCOM C/386 ▶High C 386 ▶Microsoft アセンブラ	▶Lattice C ▶Microsoft C ▶Microsoft Fortran ▶Microsoft アセンブラ
動作環境	▶MS-DOS V3.1以降 ▶MS-WINDOWS 3.0以降	

価格表	シングル・ライセンス	コーポレート・ライセンス
DOS/4G	¥880,000	¥2,640,000
DOS/4G用VMM	¥580,000	¥1,740,000
DOS/16M	¥780,000	¥2,340,000
DOS/16M用VMM	¥480,000	¥1,440,000

株式会社 LIFEBOAT
〒101 東京都千代田区神田錦町3-6
PHONE:03-3293-4711㈹ FAX:03-3293-4710

※複数マシン(最大20台)で開発を行う際にお得なコーポレート用ライセンスも用意しております。

※製品についてのお問い合わせは……㈱LIFEBOAT 営業1課 DOSエクステンダー担当(Tel.03-3293-4714)までご連絡ください。

● Copyright Notice ――― ＊DOS/4G, DOS/16MはRational Systems, Inc.の商標です ＊MS-DOS, MSはMicrosoft Corp.の登録商標です ＊その他、プログラム名、システム名、CPU名は一般に各メーカーの登録商標です。

MITSUBISHI

技術がつくる高度なふれあい SOCIO-TECH

[32bitでノート形、だからビジネスがスピーディ。]

「楽勝か、惜敗か。」は判断の速さで決まる。「楽勝研究会」の選ぶパソコンは「MAXYNOTE386」。32bit CPU搭載で、さまざまな情報を高速処理。交換自在のハードディスクパック方式で、大容量情報を自由自在に操れる。だから、ビジネスがスピーディだ。コンピュータと通信の高度な技術でニーズにこたえる、三菱情報通信システム。ビジネス「楽勝」のヒケツです。

- ●32bitCPU386™SX搭載。ビジネスを高速処理。
- ●ワンタッチで交換自在、高速&大容量ハードディスク内蔵。
- ●多彩なシステムアップ能力、MAXYSTATIONⅡ。

三菱AXパソコン

MAXYNOTE386

■標準価格…398,000円(税別：HDD 20MBモデル)

Hiroyuki Sanada

◎スピードの遅いやつは、国際ビジネスについてけない。

ニーズから発想する三菱情報通信システム
Think&Link

386™SXは米国インテル社、MAXY、MAXYSTATIONは三菱電機㈱の登録商標です。
MAXYメニューシステムは、㈱エージーソフトのAZ-MOS(エージーモス)をベースとしたメニューシステムです。AZ-MOSは㈱エージーソフト㈱アイシーエムが開発したインストール方式「IBFファイル(インストールバッチファイル)システム」を採用しています。
世界標準化を推進するAX。日本のソフトウェアメーカ、ハードウェアメーカ、流通各社によるAX協議会★のもと、OS、キーボード、画面、プリンタ部を規格化。国内・海外の豊富なソフトや周辺機器の共通利用を実現します。★1991年7月末現在参加約607社

三菱電機株式会社

お問合せは…三菱電機株式会社 〒100 東京都千代田区丸の内2-2-3(三菱電機ビル) 本社情報処理機器事業部計画部(03)3218-3285 本社情報処理機器部(03)3218-3515 本社システム営業第一部(03)3218-3547 本社システム営業第二部(03)3218-3298 本社情報通信システム計画部(03)3218-3236 本社電機事業部(03)3218-3158 ●札幌(011)212-3739 ●仙台(022)264-5656 ●大宮(048)653-0261 ●千葉(0472)41-8671 ●横浜(045)212-3521 ●新潟(025)241-7220 ●金沢(0762)52-8821 ●名古屋(052)565-3246 ●大阪(06)347-2421 ●広島(082)248-5330 ●高松(0878)25-0009 ●福岡(092)721-2346

カタログ請求券 AXパソコン プログラマーズ 9112 あなたの電話番号を記入ください。

T4911777712989

〈資料請求番号P04〉

雑誌コード17777-12